CAPÍTULO 4 – NÍVEL DE LÓGICA DIGITAL

4.1 Introdução

- O Nível de lógica digital é o nível mais baixo da Arquitetura.
- Responsável pela interpretação de instruções do nível superior (Macroinstruções - Nível ISA, ou Microinstuções - Nível de Microprogramação).
- Equivalência entre *hardware* e *software*: funções desempenhadas pelo *software* podem migrar para o *hardware* e vice-versa.
- Constituído pelos circuitos lógicos digitais (*hardware* da máquina).
- Projeto em nível de transferência entre registradores: a arquitetura pode ser entendida como um conjunto de registradores, associados a módulos lógicos adicionais para processamento de informação, interconectados de maneira apropriada.
- O seqüenciamento adequado dos sinais de controle destes módulos e registradores produz o fluxo ordenado de informação entre os mesmos, necessário à interpretação de instruções do nível superior.

4.2 Módulos Lógicos Digitais Básicos

- Os circuitos lógicos são construídos, de acordo com a Álgebra de Boole, a partir da combinação das portas lógicas básicas: AND, OR, NOT; bem como das suas combinações: NAND, NOR, XOR.
- Um único tipo de porta, NAND ou NOR, é suficiente para construir qualquer circuito lógico.
- Fisicamente, as portas são implementadas através de transistores integrados em pastilhas (*chips*) de silício. De acordo com a densidade de componentes, os *chips* são classificados em:
  - Integração em Pequena Escala (SSI): de 1 a 10 portas lógicas.
  - Integração em Média Escala (MSI): de 10 a 100 portas lógicas.
  - Integração em Grande Escala (LSI): de 100 a 100.000 portas lógicas.
  - Integração em Escala Muito Grande (VLSI): acima de 100.000 portas lógicas

- Decodificador:
  - Entrada: código de n bits, C_{n-1}, ..., C_1, C_0.
  - m = 2^n linhas de saída, $S_{(2^n-1)}$, ..., S_1, S_0.
  - Cada linha de saída é associada exclusivamente a uma palavra de código correspondente.
  - Uma palavra de código de entrada ativa apenas a linha de saída associada. As outras permanecem desativadas.
  - Exemplo de aplicação: seleção de um dentre vários *chips*.

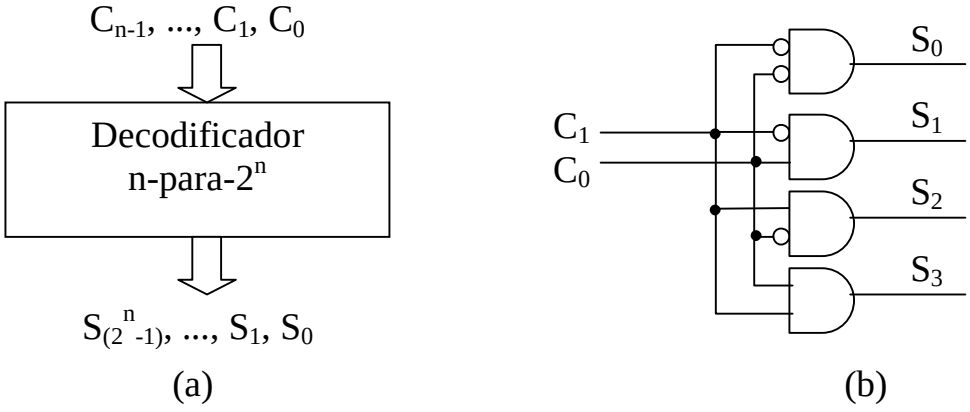


Figura 4.1. a) Decodificador. b) Exemplo de decodificador 2-para-4.

- Multiplexador:
  - Entradas: m = 2^n linhas de entrada, D_{n-1}, ..., D_1, D_0.
  - Entradas de controle código de n bits C_{n-1}, ..., C_1, C_0.. Seleciona uma das 2^n entradas.
  - Saída, S: assume o estado lógico da entrada selecionada pelo código.
  - Exemplo de aplicação: transferência de vários barramentos de entrada para um barramento de saída.

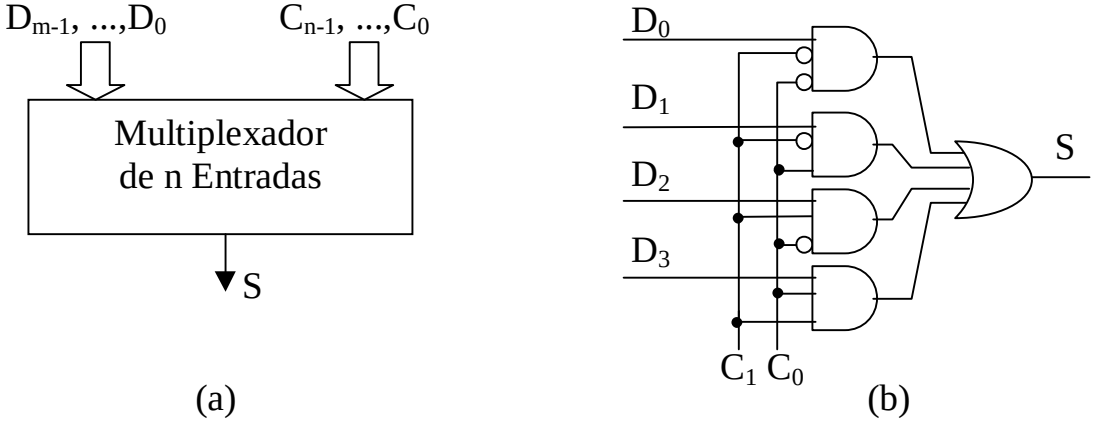


Figura 4.2. a) Multiplexador. b) Exemplo de multiplexador de 4 entradas.

- Comparador:
  - Entradas: duas palavras de n bits, A = A_{n-1}, .., A_0 e B = B_{n-1}, .., B_0.
  - Saída, S = 1, se A = B, S = 0, caso contrário.
  - Exemplo de aplicação: na ULA, para ativar bits do registrador de STATUS.

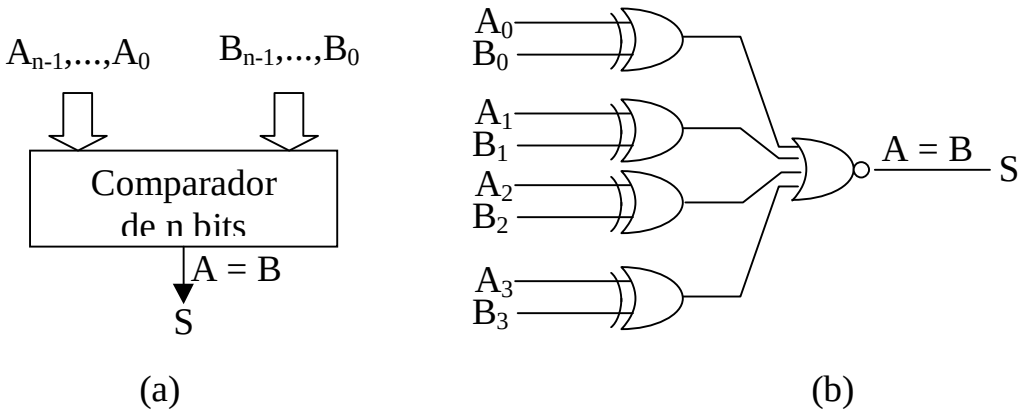


Figura 4.3. a) Comparador de n bits. b) Comparador de quatro bits.

- Deslocador:
  - Entrada: uma palavra de n bits, D = D_{n-1}, .., D_0.
  - Entrada de controle C: define direção de deslocamento. Exemplo: deslocamento à direita C = 1, deslocamento à esquerda C = 0.
  - Saída: uma palavra S = S_{n-1}, .., S_0 correspondente à palavra de entrada deslocada de um bit à esquerda ou à direita de acordo com C.

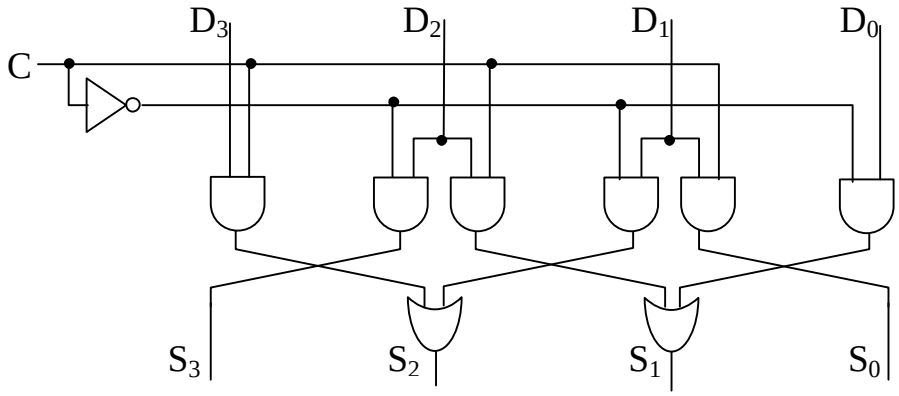
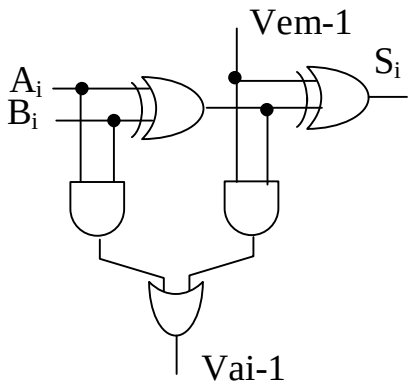


Figura 4.4. Deslocador de um bit à Direita/Esquerda – palavra de 4 bits.

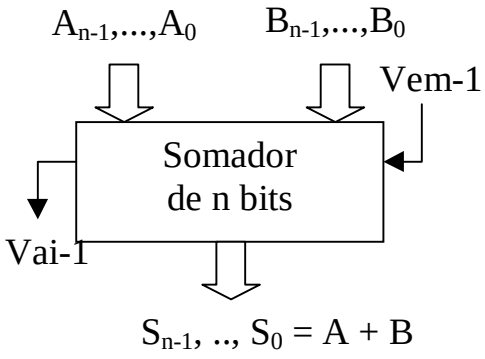
- Somador:
  - Entradas: duas palavras de n bits, A = A_{n-1}, .., A_0 e B = B_{n-1}, .., B_0.
  - Saída: uma palavra S = S_{n-1}, .., S_0 = soma binária de A e B.
  - Entrada auxiliar: Vem-1, proveniente de outro somador de ordem menos significativa.
  - Saída auxiliar: Vai-1, para outro somador de ordem mais significativa.



(a)

A	B	Vem-1	S	Vai-1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b)



(c)

Figura 4.5. a) Somador Completo de 1 bit. b) Tabela verdade correspondente. c) Somador de n bits.

4.3 Memória Principal

- Relógio:
  - Circuito baseado em cristal, que gera trem de pulsos de freqüência constante, para sincronizar os circuitos lógicos do computador garantindo a ordem de eventos, compensando atrasos diversos nas portas lógicas.
  - 1 ciclo de relógio = 1 ciclo de máquina.
  - Subciclos = temporização do ciclo de máquina. (Em máquinas microprogramadas, temporiza a microinstrução).

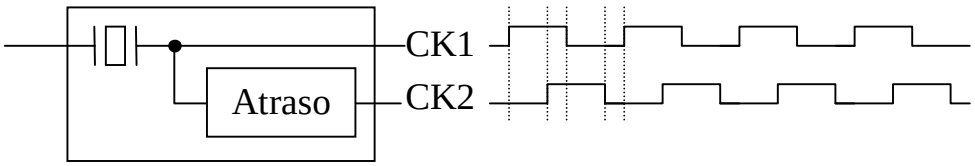
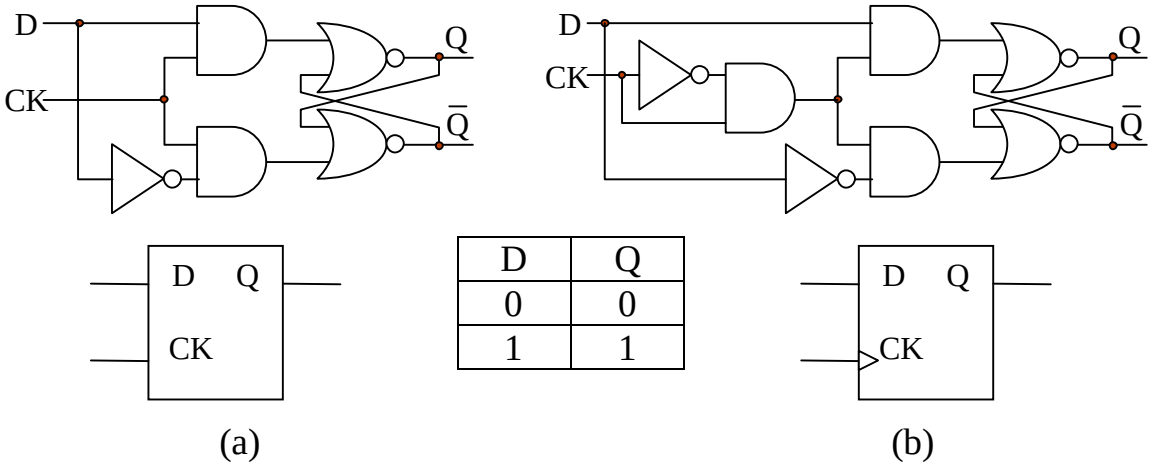


Figura 4.6. Divisão do relógio em subciclos.

- *Latches* e Flip-Flops:
  - *Latch*: Dispositivo bi-estável. Unidade de armazenamento. armazena um bit de informação. Sensível ao nível do relógio.
  - Flip-flop: Dispositivo bi-estável. Unidade de armazenamento. armazena um bit de informação. Sensível à transição do relógio.



D	Q
0	0
1	1

Figura 4.7. a) *Latch* tipo D. b) Flip-Flop tipo D.

- Registradores:

  - Registrador de n bits é um conjunto de n *latches* ou flip-flops com linhas de controle comum (CK, CLR, PR, OE, etc.).
  - Armazena uma palavra de n bits.
  - Número de pinos linearmente proporcional à capacidade de armazenamento n.

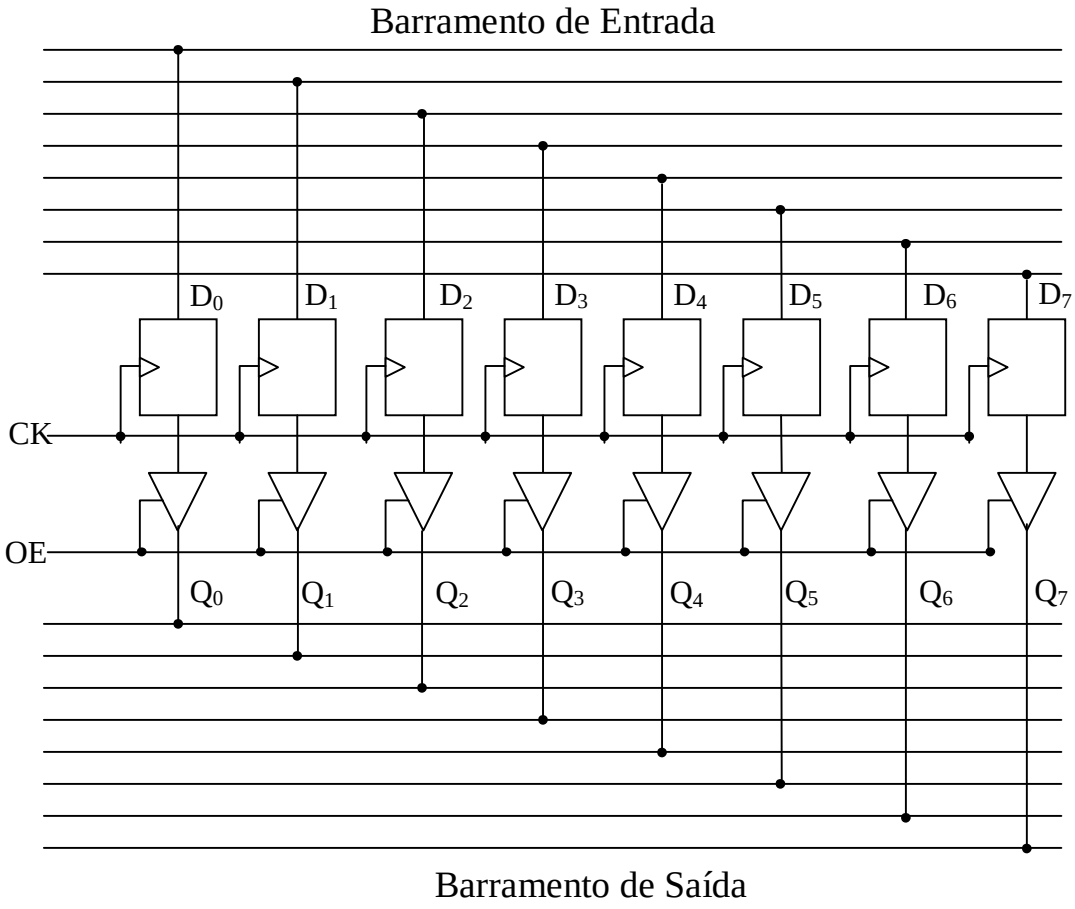


Figura 4.8. Registrador de oito bits.

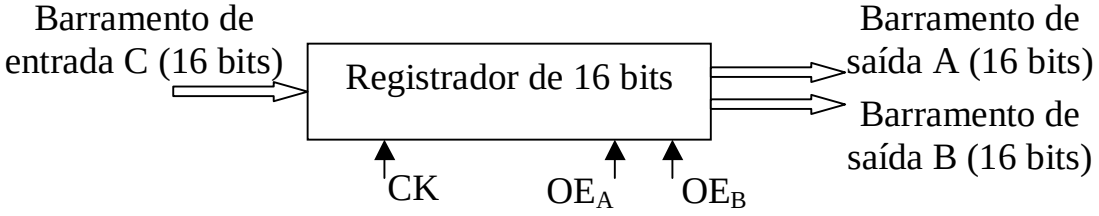


Figura 4.9. Exemplo de registrador de 16 bits com um barramento de entrada e dois barramentos de saída.

- Memória:

  - Uma memória de m x n bits é um conjunto de m registradores de n bits cada um, compartilhando linhas de dados e linhas de controle comuns.
  - Capacidade de armazenamento = m x n bits = m x n/8 bytes.
  - Cada registrador de n bits é denominado célula ou posição de memória.
  - Numa referência à memória, apenas uma célula de m bits pode ser acessada.
  - Células a serem referenciadas são selecionadas através de um código de k bits (endereço), A_{k-1}, ..., A_1, A_0.
  - $k = \log_2 m \Rightarrow$ O número de pinos do *chip* de memória cresce com o logaritmo da capacidade de armazenamento.
  - Por uma questão de eficiência na representação binária do endereço, o número de células m é escolhido como uma potência inteira de 2.
  - Os pinos de um *chip* de memória podem ser classificados em três grandes grupos:
    - Pinos de dados: D_{n-1}, ..., D_1, D_0. Permitem a transferência em paralelo de uma palavra de n bits de ou para uma célula.
    - Pinos de endereços: A_{k-1}, ..., A_1, A_0. Permitem a seleção de uma única célula dentre as 2^k disponíveis para referência.
    - Pinos de controle: (Exemplo: CS, RD, OE, etc.). Comandam a transferência de dados.
  - Os pinos de dados, endereços e controle estão diretamente ligados, respectivamente, às vias de dados, endereços e controle correspondentes do barramento.

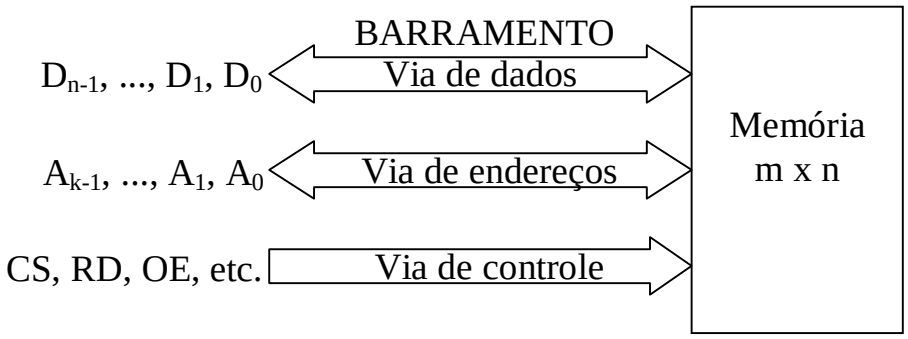


Figura 4.10. Sinais lógicos em um *Chip* de Memória de m x n bits

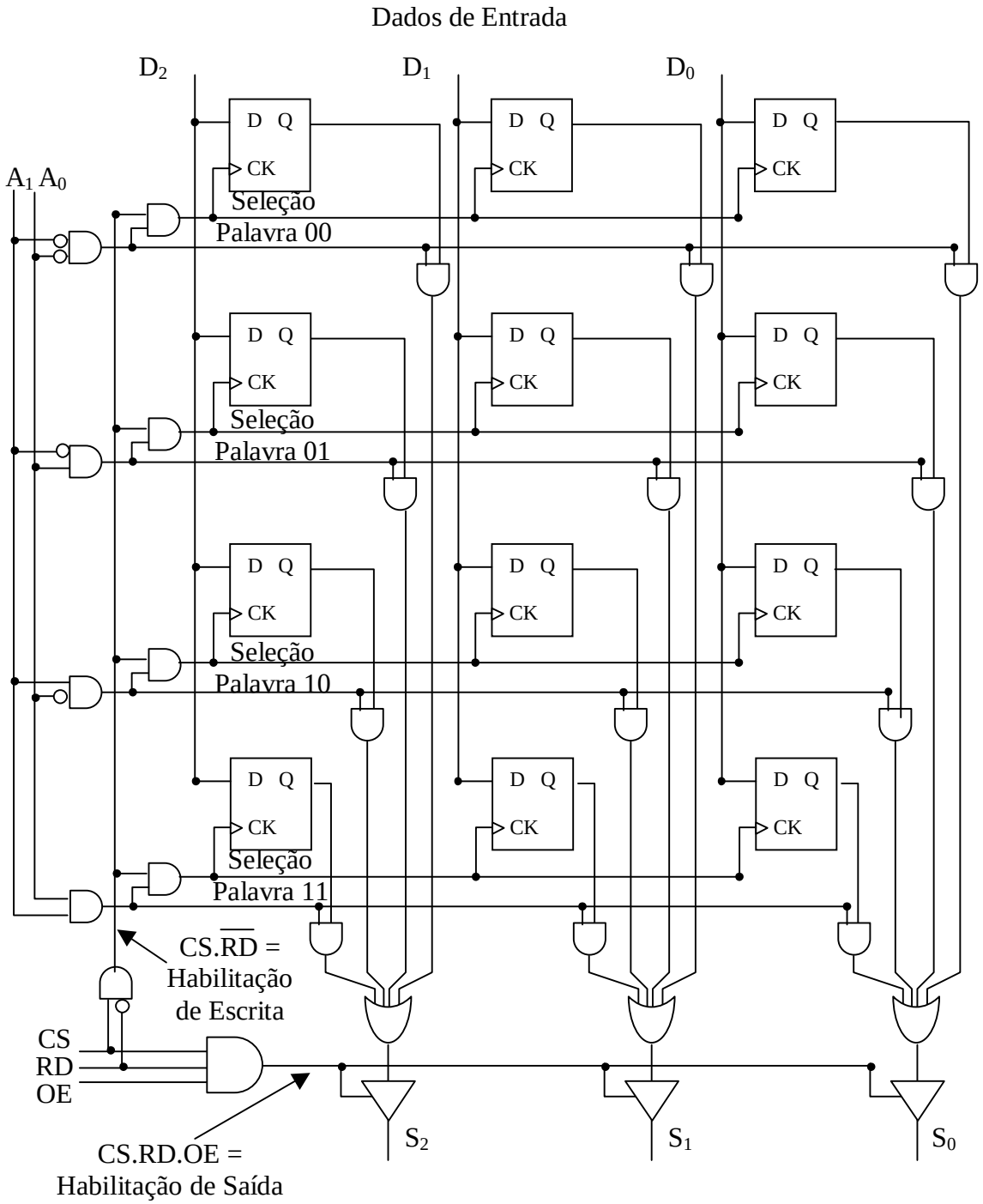


Figura 4.11. Organização de uma Memória RAM de 4 palavras de 3 bits.

- Exemplo: Memória RAM de 4 palavras de 3 bits.
  - Via de dados: transferência de dados entre a memória e outros dispositivos.
    - Entrada: $D_2D_1D_0$.
    - Saída: $S_2S_1S_0$. (*Tri-State*).
  - Via de Endereços: seleção de palavra a ser referenciada.
    - A_1A_0 = (00,01,10,11) = (palavra 0,palavra 1,palavra 2,palavra 3).
  - Via de Controle: habilitam a referência à palavra endereçada.
    - CS (*Chip Select*): 1 seleciona o *chip* para qualquer operação.
      0 não seleciona o *chip*.
    - RD (*Read*): 1 seleciona operação de leitura.
      0 seleciona operação de escrita.
    - OE (*Output Enable*): Habilita saída. *Buffer* não inversor *tri-state* permite desconectar logicamente a memória do barramento através de conexão de alta impedância.

- Tipos de Memórias Semicondutoras:

  - SRAM (*Static Random Access Memory*):
    - Memória de Leitura/Escrita, volátil, apagada eletricamente.
    - Baseada em flip-flops, que ocupam um espaço razoável no *chip*.
    - Mantém o conteúdo indefinidamente desde que a alimentação elétrica seja mantida.
    - Muito rápidas (nanossegundos), usadas em memória Cache.
  - DRAM (*Dynamic Random Access Memory*):
    - Memória de Leitura/Escrita, volátil, apagada eletricamente.
    - Baseadas em capacitores, que ocupam pouco espaço no *chip*, garantindo uma alta densidade de bits/cm^2 a baixo custo.
    - Necessitam restaurar periodicamente o conteúdo (*refresh*).
    - Lentas (dezenas de nanossegundos).
    - Aplicação típica: memória principal.
  - DRAM FPM (*DRAM Fast Page Mode*):
    - Memória DRAM organizada como uma matriz de bits.
    - Endereçada em duas etapa: por linha (RAS - *Row Address Strobe*) e por coluna (CAS - *Column Address Strobe*). Mais lentas.
  - DRAM EDO (*DRAM Extended Data Output*):
    - Construída dentro dos princípios do *pipeline*. Permite começar uma referência antes do fim da referência anterior.
    - Melhora a banda passante (bits/s) da memória.

- SDRAM (*Synchronous DRAM*):
  - Linhas de dados e endereços sincronizadas por sinal de *clock*.
  - Aplicação típica: Caches de maior capacidade.
- ROM (*Read Only Memory*):
  - Memória apenas de leitura, não volátil.
  - Conteúdo não pode ser alterado, gravada durante a sua fabricação.
  - Baixo custo.
  - Aplicação típica: eletrônica de consumo (grandes volumes).
- PROM (*Programmable ROM*):
  - Memória apenas de leitura, não volátil.
  - Gravada pelo usuário, uma única vez. Não é alterável por byte.
  - Equipamentos produzidos em pequeno volume.
- EPROM (*Erasable PROM*):
  - Memória apenas de leitura, não volátil.
  - Gravada pelo usuário. Reprogramável. Não é alterável por byte.
  - Conteúdo pode ser apagado através de radiação ultravioleta (UVEPROM).
  - Aplicação típica: desenvolvimento de protótipos.
- EEPROM (*Electrically Erasable PROM*):
  - Memória apenas de leitura, não volátil.
  - Gravada pelo usuário. Reprogramável. É alterável por byte.
  - Conteúdo pode ser alterado através de sinais elétricos, sem precisar remover o *chip* do circuito.
  - Menor capacidade e velocidade 50 % menor do que as memórias EPROM.
  - Muito mais caras, com capacidade cem vezes menor e dez vezes mais lentas do que as memórias RAM.
  - Aplicação típica: desenvolvimento de protótipos.
- FLASH:
  - Memória de leitura/escrita, não volátil.
  - Conteúdo pode ser alterado através de sinais elétricos, sem precisar remover o *chip* do circuito.
  - Não é alterável por byte. Apagada e regravada em blocos.
  - Tempo de aceso típico: 100 nanossegundos.
  - Não suportam muitas regravações (em torno de 10.000).
  - Aplicação típica: câmeras digitais.

4.4 Barramento

- Barramento: caminho de dados compartilhado por vários dispositivos. Conjunto de conexões em paralelo, através das quais os dados são transmitidos entre os dispositivos conectados ao barramento.
- Protocolo de barramento: regras de funcionamento do barramento que devem ser seguidas por todos os dispositivos a ele conectados. Especificações lógicas, elétricas e mecânicas.

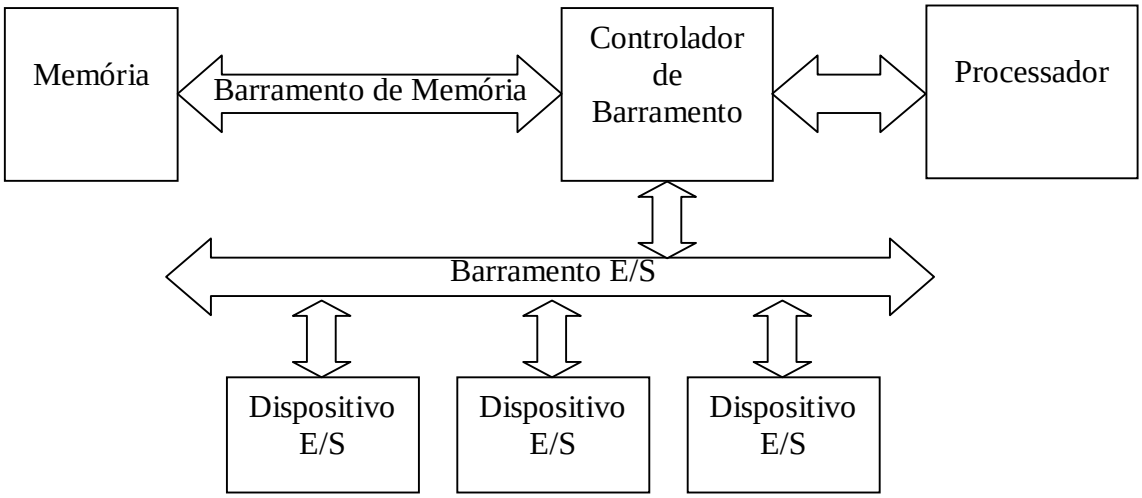


Figura 4.12. Barramentos típicos em um computador.

- Mestre de barramento: dispositivo que inicia uma transferência, agindo ativamente na mesma.
- Escravo de Barramento: dispositivo que, numa transferência, responde passivamente às requisições de um mestre.
  - Observação: um mesmo dispositivo pode atuar ora como mestre, ora como escravo. Memórias sempre atuam como escravos.
- Para adequar os sinais usados nos dispositivos aos sinais usados no barramento (e vice-versa), utilizam-se circuitos de interfaceamento (*chips*) adequados: alimentadores de barramento, (*bus drivers*), para os mestres e receptores de barramento, (*bus receivers*), para os escravos.
- *Bus Driver* + *Bus Receiver* = *Bus Tranceiver*, usados por dispositivos que tanto podem assumir o papel de mestre como o de escravo.
- Conexão com o barramento:
  - *Tri-state*.
  - Coletor aberto (*Wired-OR*).

- Largura do Barramento:

  - Via de Controle: Transmite sinais de controle diversos para controlar o tráfego de informação pelo barramento (linhas de arbitragem, linhas de interrupção, linhas de controle de leitura e escrita, *clock*, *Reset*, etc.).
  - Via de Endereços: Transmite os bits de endereço. O número de linhas determina o número de bits usados para endereçar memória principal e, consequentemente, o tamanho do espaço de endereçamento. n linhas $\Rightarrow 2^n$ posições de memória.
  - Via de Dados: Transmite os dados a serem transferidos propriamente ditos. Quanto mais larga, mais dados são transmitidos em paralelo, consequentemente, maior a banda passante (bps).

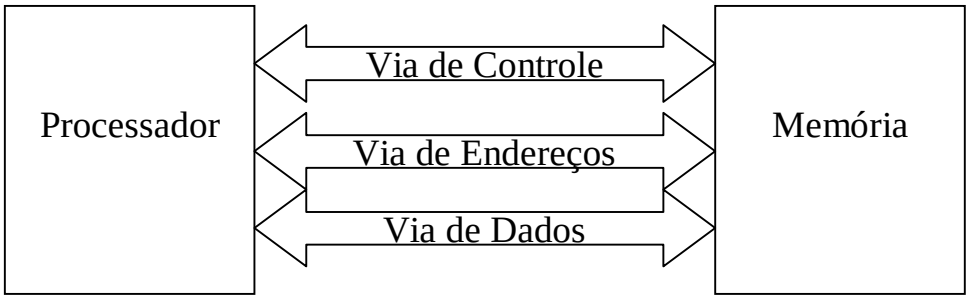
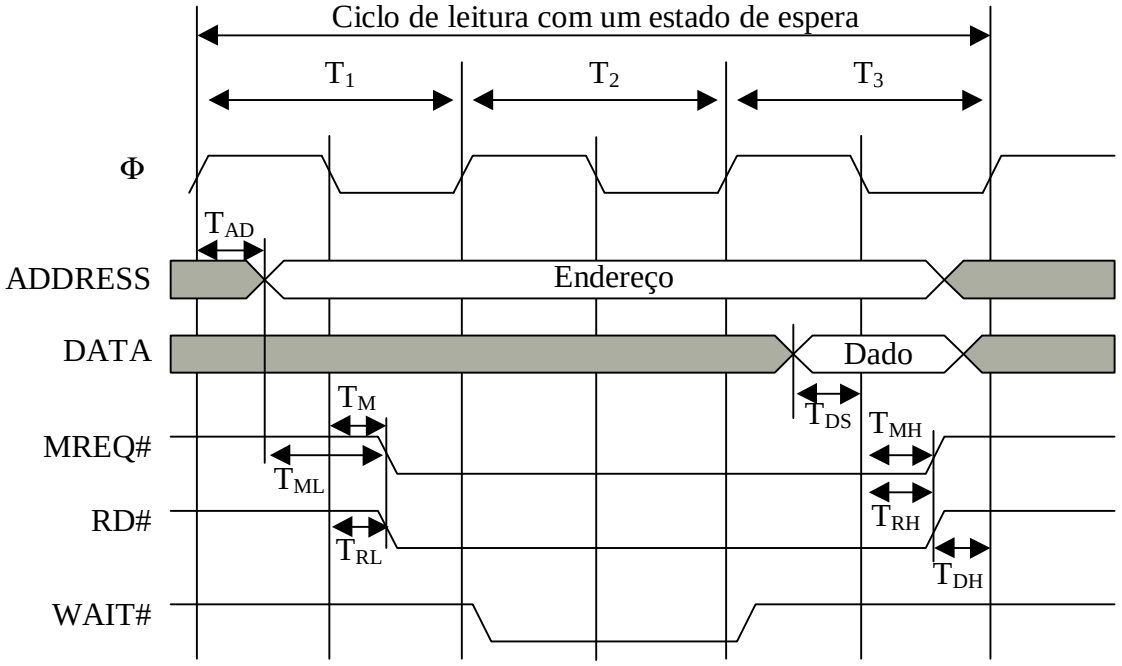


Figura 4.13. Vias de Controle, Endereços e Dados em um barramento.

- Quanto mais "largo" o barramento, (mais linhas em paralelo), maior será a banda passante, mas também, maior o espaço físico ocupado na placa e maiores os conectores, de modo que as linhas precisam ser cada vez mais estreitas.
- Alternativa para aumentar a banda passante sem aumentar a largura: aumentar a velocidade do barramento. Problema: "escorregamento" dos sinais (*bus skew*), que consiste em atrasos de propagação diferentes em linhas diferentes, pode tornar-se significativo, comprometendo o sistema.
- Alternativa para aumentar a largura sem aumentar o número de linhas: Barramento Multiplexado. Mesma via utilizada para transmitir endereços (no inicio de uma transferência) e para transmitir dados (após o endereçamento ter sido consumado). Resulta em custo menor às custas de um tempo de transferência maior.

- Temporização do Barramento:

  - Barramento Síncrono:
    - Possui um relógio mestre (oscilador a cristal que gera uma onda quadrada de freqüência estável).
    - Todas as operações do barramento são sincronizadas pelo relógio.
    - Funciona "em malha aberta". Não há interação direta entre mestre e escravo.
    - Todas as operações levam um número inteiro de ciclos do relógio (Ciclos do Barramento).
    - Vantagem: barato e de fácil implementação. Muito utilizado.
    - Desvantagem: pares mestre-escravo mais rápidos ficam limitados ao ciclo do barramento.
    - Exemplo de operação de leitura:
      - Características do sistema: Barramento de 40 MHz (ciclo de 25 ns). Tempo de leitura da memória após o endereço estabilizar no barramento igual a 40 ns. Três ciclos para a operação de leitura.
      - Sinais de controle (observação: # denota ativo baixo):
        - Φ = relógio.
        - ADDRESS: sinais de endereçamento.
        - DATA = dados a transferir.
        - MREQ#: requisição de referência à memória.
        - RD#: ativo = leitura, inativo = escrita.
        - WAIT# = sinaliza estado de espera até que a memória consiga atender à requisição.
      - Operação:
        - A operação é disparada pela transição positiva do relógio no primeiro ciclo T_1.
        - O processador coloca o endereço na via de endereços.
        - Após estabilização de ADDRESS, disparado pela transição negativa de T_1, o processador ativa MREQ# e RD#.
        - No início de T_2, a memória ativa WAIT#, sinalizando que não tem condições de fornecer o dado neste ciclo.
        - Na transição positiva de T_3, a memória desativa WAIT#, sinalizando que o dado será disponibilizado neste ciclo. A memória coloca o dado DATA na via de dados.
        - Na transição negativa de T_3 o processador lê o dado no barramento, desativando MREQ# e RD#.



Símbolo	Parâmetro	Mínimo	Máximo
T_{AD}	Retardo de endereço		11 ns
T_{ML}	Endereço estável antes de MREQ#	6 ns	
T_M	Retardo de MREQ# a partir da transição negativa do *clock* em T_1		8 ns
T_{RL}	Retardo de RD# a partir da transição negativa do *clock* em T_1		8 ns
T_{DS}	Tempo de estabelecimento antes da transição negativa do *clock* em T_3	5 ns	
T_{MH}	Retardo de MREQ# a partir da transição negativa do *clock* em T_3		8 ns
T_{RH}	Retardo de RD# a partir da transição negativa do *clock* em T_3		8 ns
T_{DH}	Tempo de permanência antes da desativação de RD#	0 ns	

Figura 4.14. Exemplo de temporização de leitura em barramento síncrono.

- O Processador deve gerar endereços, no máximo, T_{AD} = 11 ns após a transição positiva de T_1.
- O dado deve estar disponível, no mínimo, T_{AD} = 5 ns antes da transição negativa de T_3.
- No pior caso, a partir do aparecimento do endereço, a memória dispõe de $(T_1+T_2+T_3/2)$ - T_{AD} - T_{DS} = 46,5 ns, para liberar o dado a ser lido.
- A memória terá somente $(T_2/2+T_3/2)$ - T_M - T_{DS} = 37 ns, para colocar o dado no barramento após a ativação de MREQ# e RD#.

- Barramento Assíncrono:

  - Não possui um relógio mestre.
  - As operações podem levar o tempo que for necessário para serem realizadas e não um número inteiro de ciclos de relógio.
  - Vantagem: suporta pares mestre-escravo heterogêneos (lentos e rápidos).
  - Funciona em malha fechada a partir de um protocolo denominado "Aperto de Mão" completo (*hand-shake* completo). Os sinais de controle gerados pelo escravo são disparados em resposta sinais gerados pelo mestre, e vice-versa.
  - Desvantagem: é mais difícil de implementar. Pouco utilizado.
  - Exemplo de operação de leitura com *hand-shake* completo
    - Sinais de controle (observação: # denota ativo baixo):
      - MSYN#: *Master SYNcrhonization*.
      - SSYN#: *Slave SYNcrhonization*.
    - Operação:
      - Após colocar o endereço no barramento e ativar os sinais MREQ# e RD#, o mestre ativa MSYN#.
      - EM resposta a MSYN#, levando o tempo que for necessário, o escravo coloca o dado requerido no barramento e ativa SSYN#.
      - Em resposta a SSYN#, levando o tempo que for necessário, o mestre lê o dado e, a seguir, desativa MSYN#.
      - Em resposta, o escravo desativa SSYN#.

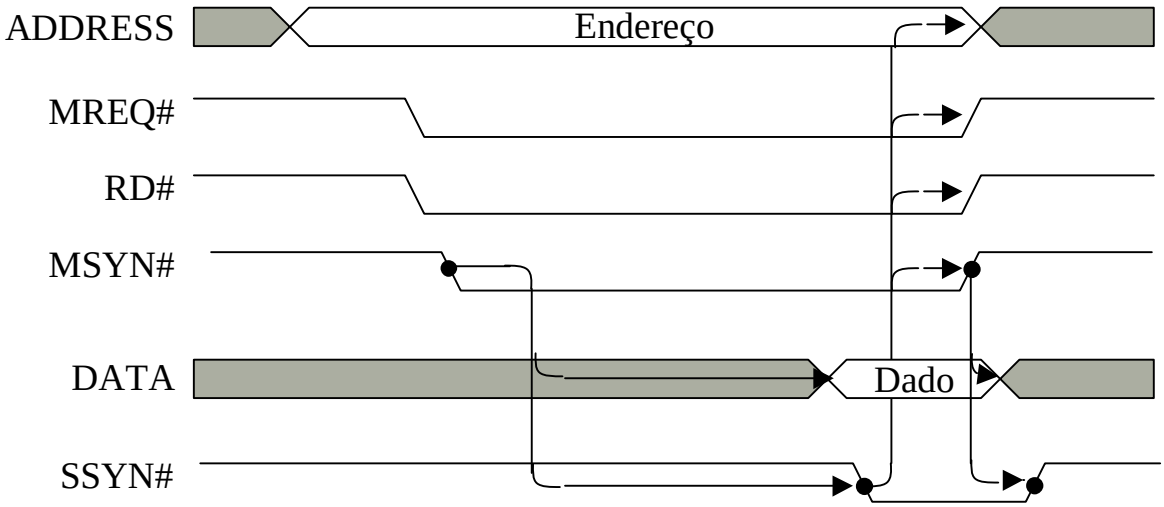
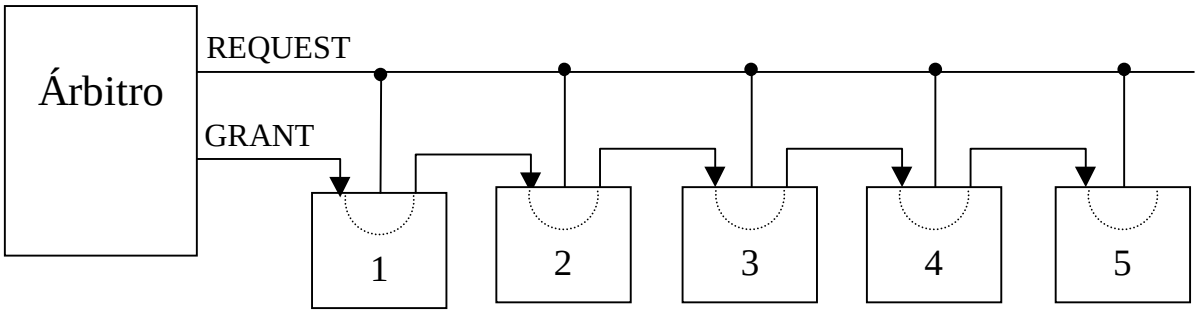


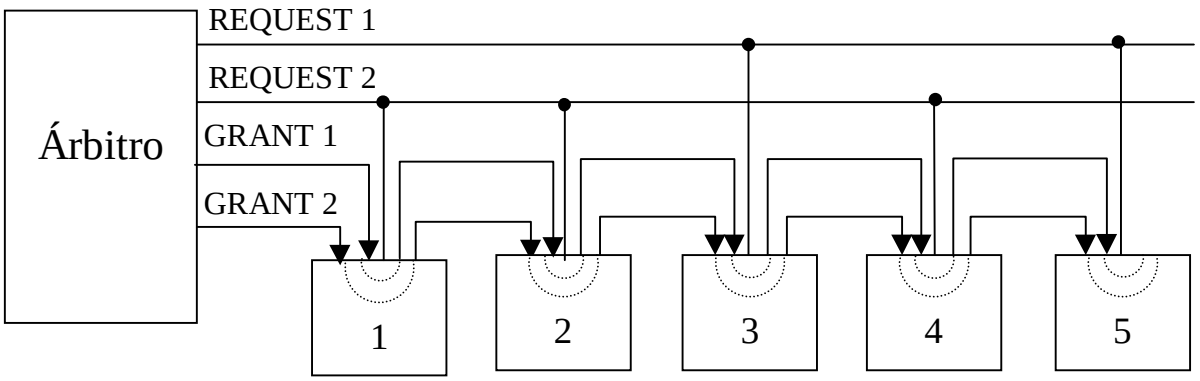
Figura 4.15. *Hand-Shake* Completo em um Barramento Assíncrono.

- Arbitragem de Barramento:

  - Arbitragem: mecanismo de seleção do mestre do barramento quando dois ou mais dispositivos desejam tomar posse do mesmo simultaneamente.
  - Arbitragem Centralizada:
    - Baseada na figura do Árbitro do Barramento, circuito lógico responsável pelo procedimento de arbitragem.
    - O árbitro, fisicamente, pode estar localizado no próprio processador ou em outro *chip* dedicado a esta função.
    - Esquema *Daisy Chaining*:
      - Dispositivos fazem requisição através de linha *wired-OR*, à qual todos estão conectados e que é ligada ao árbitro. A linha é ativada quando um ou mais dispositivos requisitam o barramento (OR lógico dos sinais *BUS REQUEST* - requisição do barramento de todos os dispositivos)
      - O árbitro só pode distinguir a ausência ou presença de requisição. Neste caso, ativa linha de garantia de uso (*BUS GRANT* - garantia de uso do barramento), ligada em série através de todos os dispositivos.
      - A garantia de uso é passada seqüencialmente de um dispositivo para outro da cadeia, a partir do árbitro, até alcançar um que tenha feito uma requisição.
      - O dispositivo que tiver feito uma requisição e que primeiro recebe a permissão interrompe a cadeia (não repassa a permissão) e toma posse do barramento.
    - Observações:
      - A prioridade de posse é baseada na distância física ao árbitro.
      - Em sistemas em que a memória está ligada no barramento comum aos dispositivos E/S, o processador competirá pela posse na maioria dos ciclos, de modo que é aconselhável atribuir ao mesmo a mais baixa prioridade. O uso de um barramento dedicado entre memória e processador contorna este problema.
      - A rigidez imposta pelo esquema de prioridades da *Daisy Cahining* pode ser contornada usando dois ou mais pares de linhas REQUEST/GRANT, com prioridades diferentes. Neste caso, o árbitro só liberará a permissão para a linha GRANT de maior prioridade dentre os níveis que fizerem requisições simultâneas.



(a)



(b)

Figura 4.16. a) Esquema de Arbitragem *Daisy Chaining*. b) *Daisy Chaining* com dois níveis de prioridade.

- Arbitragem Descentralizada:
  - Não existe a figura do árbitro. O processo de arbitragem é distribuído entre todos os dispositivos.
  - Esquema baseado em múltiplas linhas de requisição:
    - Cada dispositivo dispõe de uma linha de requisição própria, com um determinado nível de prioridade.
    - Todas os dispositivos monitoram todas as linhas de requisição.
    - Se um determinado dispositivo fez uma requisição, saberá a todo momento se existe algum outro dispositivo com maior prioridade requisitando o barramento.
    - De acordo com a sua prioridade, o dispositivo tem condições de saber se pode ou não pode tomar posse do barramento.
    - Desvantagem: a necessidade de uma linha de requisição por dispositivo limita o número de dispositivos que podem ser conectados ao barramento.

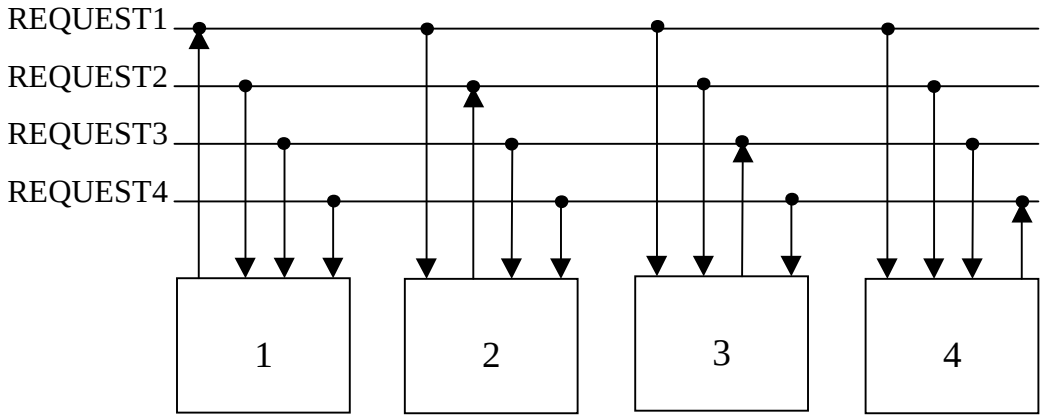


Figura 4.17. Arbitragem descentralizada com múltiplas linhas REQUEST.

- Esquema baseado em propagação de permissão:
  - Similar ao *Daisy Chaining*, mas sem árbitro centralizado.
  - Utiliza duas linhas BUSY e Arbitragem:
    - BUSY (Ocupado*)*. Mantida ativa pelo mestre atual do barramento e desativada ao finalizar uma transação.
    - Linha de arbitragem. Possui uma extremidade na tensão de alimentação. V_{CC}. O sinal de arbitragem se propaga livremente através de todos os dispositivos que não estiverem usando o barramento.
  - Para cada dispositivo, a linha de arbitragem tem um terminal de entrada (IN) e um de saída (OUT).
  - Um dispositivo que deseja se tornar mestre deve monitorar a linha BUSY até o barramento ser desocupado e verificar se o sinal de arbitragem está ativo na sua entrada IN.
  - Caso IN não estiver ativo, não pode se tornar mestre.
  - Caso IN estiver ativo, o dispositivo nega OUT. Este sinal se propagará ao longo do resto da cadeia, desativando os INs seguintes. O dispositivo torna-se mestre e ativa BUSY e OUT.
  - Observação: a prioridade depende da distância do dispositivo ao inicio da linha de arbitragem.

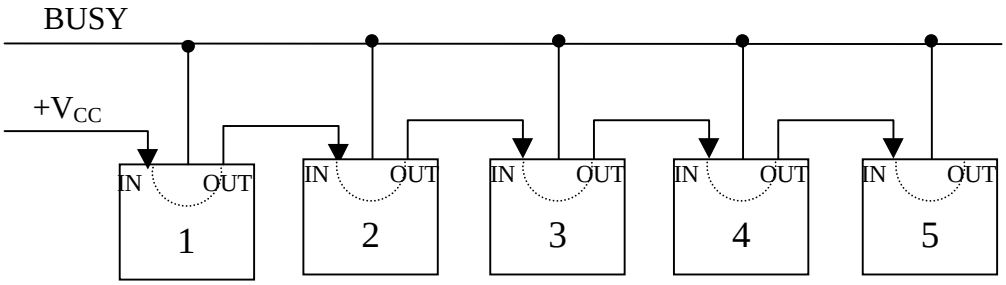


Figura 4.18. Arbitragem descentralizada com propagação de permissão.

- Outras operações no Barramento:

  - Transferência em Bloco:
    - Mais eficiente do que transmitir uma palavra por vez.
    - No primeiro ciclo da transferência, o mestre ativa um sinal de controle BLOCK# para requisitar transferência em bloco e coloca no barramento o valor de um contador que informa ao escravo o tamanho do bloco .
    - Em resposta, a cada ciclo, o escravo coloca uma palavra no barramento, decrementando o contador até o mesmo zerar.

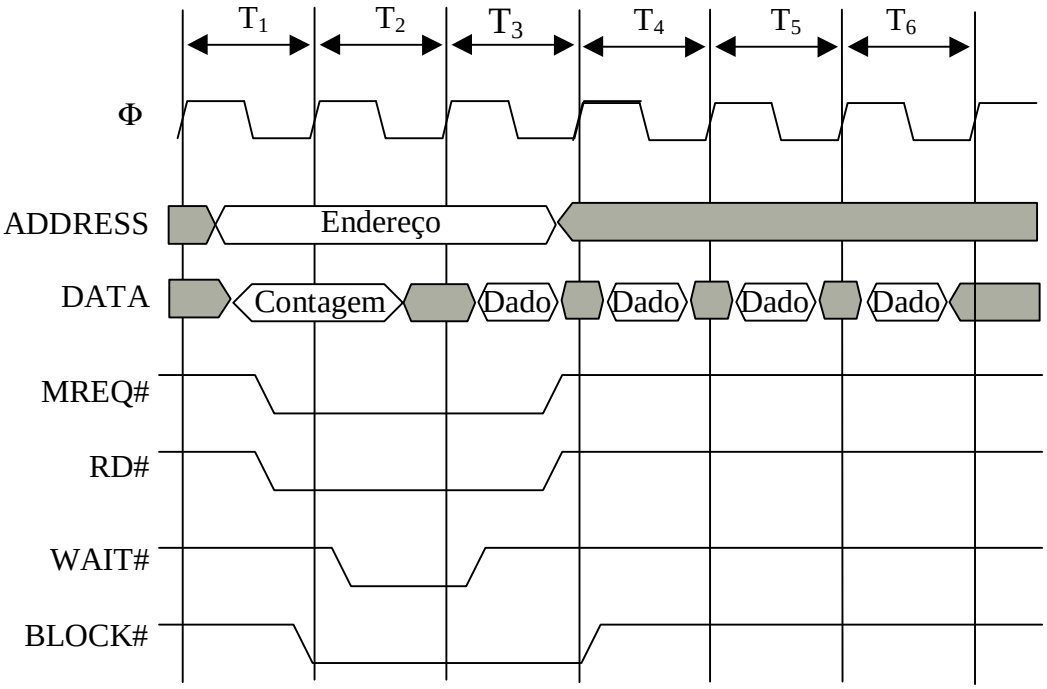


Figura 4.19. Transferência em bloco.

  - Transferência com bloqueio de leitura:
    - Em sistemas multiprocessadores, é necessário assegurar que apenas um processador de cada vez terá acesso a dados compartilhados.
    - Solução: disponibilizar ciclo especial para ler-modificar-escrever, que possibilita que, sem liberar o barramento, um processador leia uma palavra, altere o seu valor e escreva o novo valor de volta na memória sem que outro processador possa referenciar essa palavra enquanto durar a transação.

- Transferência em Interrupções:
  - Quando dois ou mais dispositivos solicitam interrupção através do barramento, é necessário um sistema de arbitragem, com atribuição de prioridades. A solução mais comum é utilizar um árbitro centralizado, um *chip* Controlador de Interrupções.
  - Exemplo: Controlador de Interrupção 8259.
    - O controlador dispõe de registradores internos que podem ser lidos ou escritos pelo processador usando sinais do barramento RD# (ReaD), WR# (Write), CS# (Chip Select), A0#, etc.
    - Os registradores internos podem ser escritos para escolher modos de operação, mascarar interrupções, etc.
    - Oito entradas de interrupção IRx recebem pedidos de interrupção de dispositivos com diferentes prioridades.
    - Quando uma ou mais entradas IRx são ativadas, o controlador ativa o sinal de interrupção do processador INT (INTerrupt).
    - Processador em condições de atender interrupção ativa sinal INTA# (INTerrupt Acknowledge) de confirmação para o controlador.
    - Controlador informa ao processador qual foi o dispositivo mais prioritário dentre aqueles que solicitaram interrupção, colocando o seu número na via de dados.
    - O processador usa este número para indexar o vetor de interrupção e, a partir deste, executar a Rotina de Serviço de Interrupção (RSI) do dispositivo correspondente. Ao fim da RSI, o processador escreve um código apropriado no controlador, de modo a avisá-lo que está pronto para atender outras interrupções.
    - Em resposta, o controlador desativa INT (desde que não exista uma outra interrupção pendente).

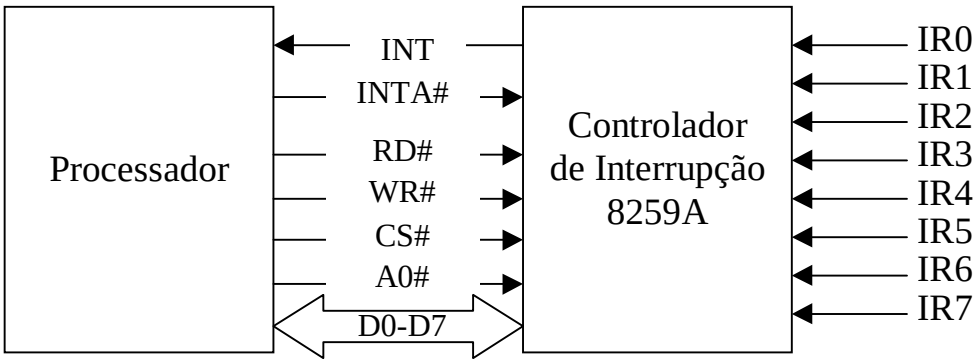


Figura 4.20. Controlador de Interrupções.

4.5 Entrada e Saída

- Entrada/Saída Paralela:
  - Módulo que permite a transferência simultânea de todos os bits de uma palavra entre um dispositivo e a via de dados do processador através de um conjunto de canais de transmissão em paralelo.

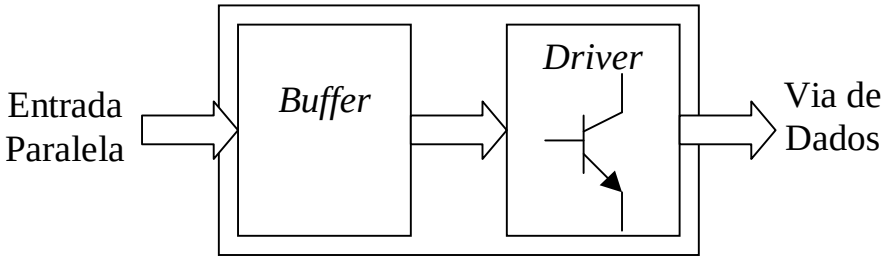


Figura 4.21. Princípio de entrada paralela de dados.

- Adaptador de Interface de Periférico (*Peripheral Interface Adaptor*):
  - Exemplo: PIA 6820 da Motorola.

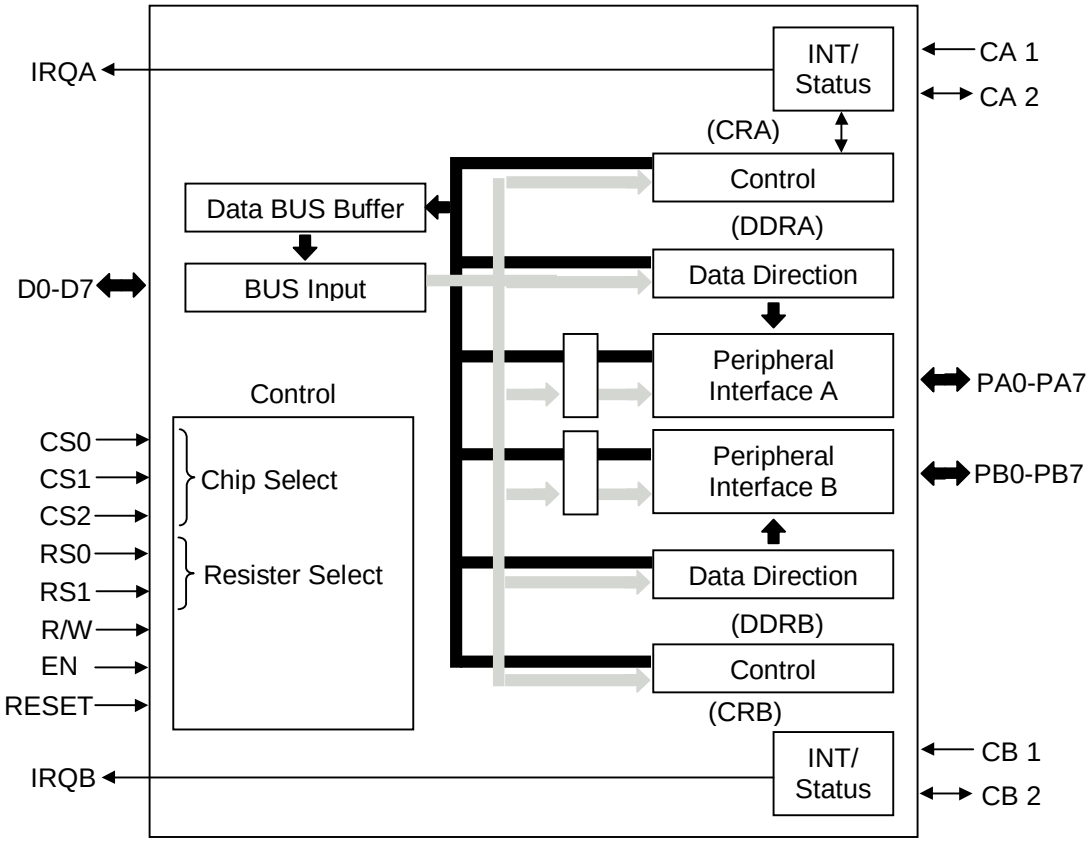


Figura 4.22. PIA 6820 da Motorola.

- Dois canais, A e B, cada um de oito bits de E/S conectados às oito linhas E/S da via de dados, D0 a D7.
- R/W especifica leitura ou escrita.
- EN condiciona o circuito de controle de interrupção interno e temporiza os sinais de controle do periférico.
- RESET inicializa o sistema.
- IRQA e IRQB: linhas de interrupção dos canais A e B.
- CA1 e CA2, CB1 e CB2: linhas de controle dos canais A e B. Sobre estas linhas troca-se informação de *status* entre computador e periférico. Exemplo: *buffer* cheio, reconhecimento de transmissão, enviar próximo byte, etc.
- Através do *Chip Select* ou *Register Select* é possível endereçar um *buffer* (que armazena temporariamente a informação transferida), bem como registradores internos.
- Os registradores de direção (*data direction*) armazenam informações de direção de transferência de dados, a qual pode ser programada *setando* ou *resetando* cada um dos bits associados a cada uma das linhas individuais de E/S.
- O registrador de controle armazena informação de comando e de *status* da transmissão (máscara de interrupção, *buffer* cheio, etc.

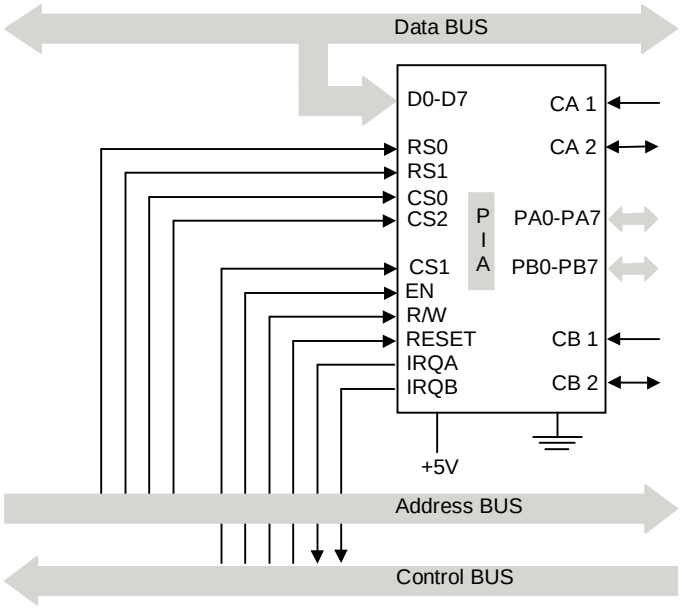


Figura 4.23. Exemplo de interface entre PIA e processador.

- Entrada/Saída Paralela (PIO - *Parallel Input/Output*):
  - Exemplo: PIO 8255A da INTEL, de propósito geral.

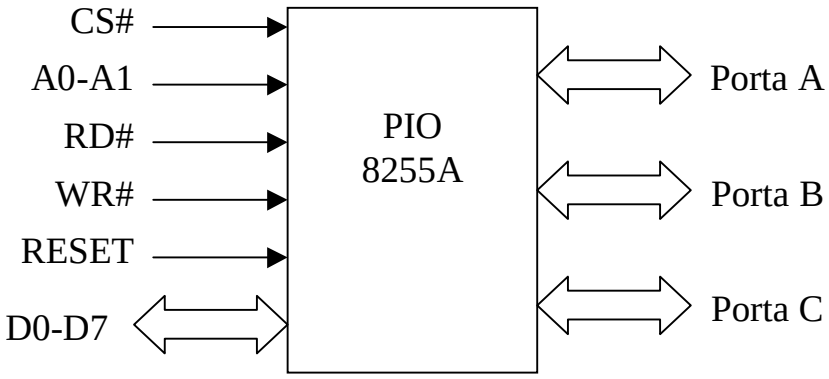


Figura 4.24. PIO 8255A da Intel.

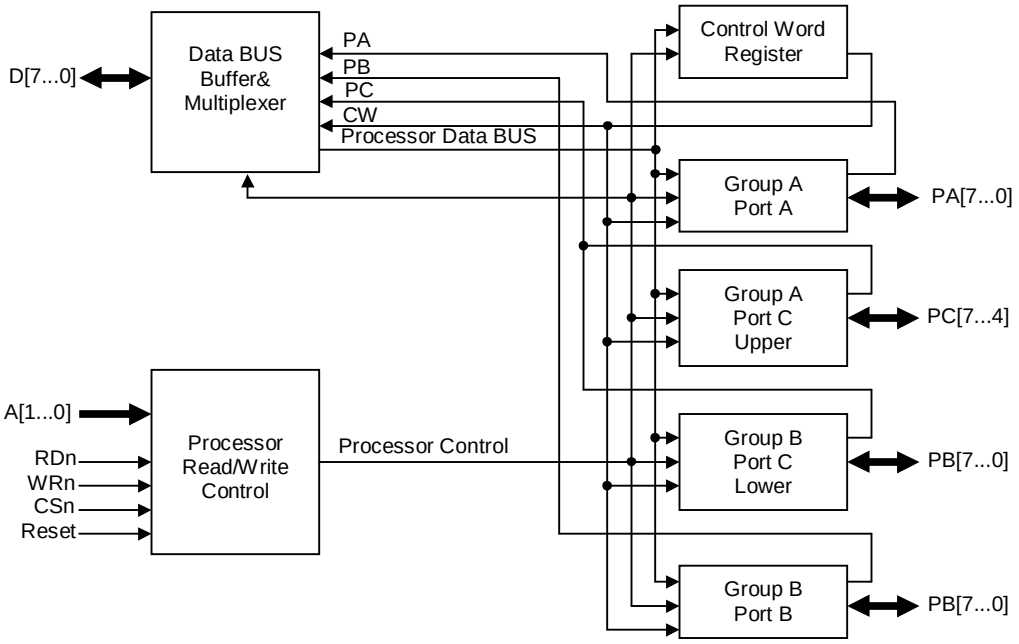


Figura 4.25. Diagrama de blocos da PIO 8255A.

- Interface com a via de dados através de *Buffer tri-sate* bidirecional de 8 bits. Por ele são transmitidos dados de entrada ou saída, bem como palavras de controle e de *status*.
- Três portas (A, B e C) configuráveis de 8 bits:
  - Porta A: um *latch/buffer* de saída de dados de 8 bits e um *latch* de entrada de dados de 8 bits.
  - Porta B: um *latch/buffer* de entrada/saída de dados de 8 bits e um *buffer* de entrada de dados de 8 bits.
  - Porta C: um *latch buffer* de entrada/saída de dados de 8 bits e um *buffer* de entrada de dados de 8 bits. Pode ser dividida em duas portas de 4 bits, que podem ser usadas para os sinais de controle de saída e *status* de entrada em conjunto com as portas A e B.
- Os 24 pinos de E/S podem ser programados individualmente em dois grupos de 12, utilizados em três modos de operação:
  - Modo 0: E/S básica - cada grupo de 12 pinos E/S pode ser programados em conjuntos de 4 como entradas ou saídas.
  - Modo 1: E/S "*Strobed*" - cada grupo de 12 pinos de E/S pode ser programado para ter oito linhas de entrada ou saída. 3 dos 4 pinos restantes são usados para *handshake* e controle de interrupção.
  - Modo 2: Barramento Bidirecional - 8 linhas são usadas como barramento bidirecional e, para *handshaking*, as 4 linhas restantes e mais uma emprestada do outro grupo de 12.
- CS# (*Chip Select*): habilita a comunicação entre o *chip* e a CPU.
- RD# - Leitura de dados. Habilita o envio do dado do *chip* à CPU através da via de dados.
- WR# - Escrita: habilita a CPU para escrever um dado ou uma palavra de controle no *chip*.
- A0 e A1 (*Port Select* 0 and *Port Select* 1): Normalmente, ligadas ao dois bits menos significativos da via de endereços. Junto com RD# e WR#, controlam a seleção de uma das três portas ou dos registradores da palavra de controle.
- Grupos de Controle A e B: recebem palavras de controle a partir do barramento e geram sinais de controle para as portas.
  - Grupo de controle A: Porta A e Porta C MSB bits (C7-C4).
  - Grupo de controle B: Porta B e Porta C LSB bits (C3-C0).

- Entrada/Saída Serial:

  - Módulo que permite a transferência dos bits de uma palavra entre um dispositivo e a via de dados do processador através de um canal serial de transmissão.
    - Na saída serial, os bits da palavra, fornecidos em paralelo a partir da via de dados, devem ser serializados e transmitidos ao dispositivo de saída, um a um, pelo canal serial.
    - Na entrada serial, os bits da palavra são enviados em série, um a um, pelo dispositivo de entrada e, recebidos a partir do canal, devem ser convertidos para uma palavra paralela a ser transferida para a via de dados.

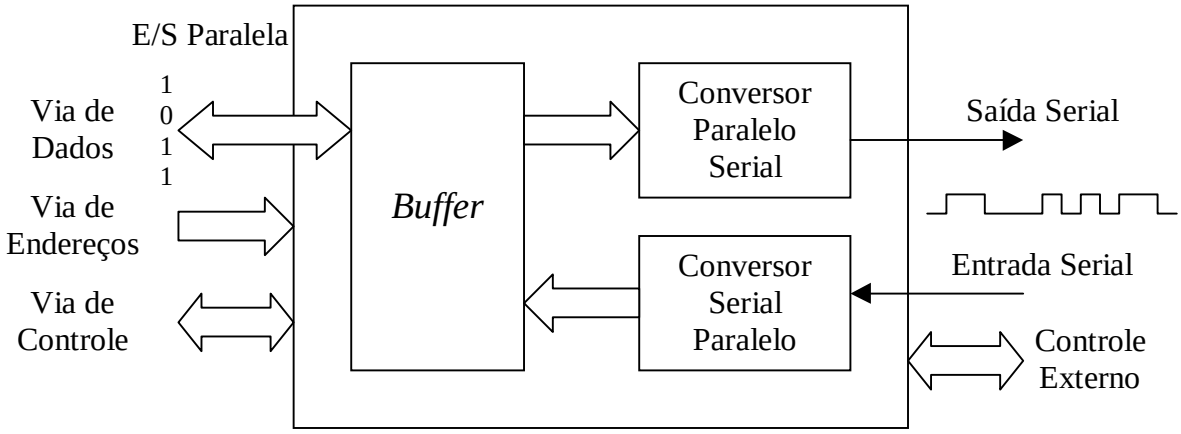


Figura 4.26. Princípio de entrada/saída serial de dados.

- USART (*Universal Synchronous/Asynchronous Receiver Transmitter*):
  - Receptor e Transmissor síncrono/assíncrono *full duplex*.
  - Exemplo: USART 8251 - Padrão IEEE para portas seriais.
  - Altamente reconfigurável.
  - Gerador de taxa de baud programável: síncrona até 64 kbaud, assíncrona até 38,4 kbaud.
  - *Buffers* de tansmissão e recepção FIFO configuráveis.
  - 5, 6, 7 ou 8 bits de dados.
  - 1 ou 2 bits de parada nos modos síncrono ou assíncrono, com 1,5 bits de parada adicionais em modo assíncrono.
  - Geração automática de paridade e outros esquemas de detecção e correção de erros.
  - Suporte de sinais de controle de Modem: CTS, RTS, DSR, DTR.
  - Composto por quatro módulos funcionais: transmissor, receptor, *buffer* de dados e unidade de controle.

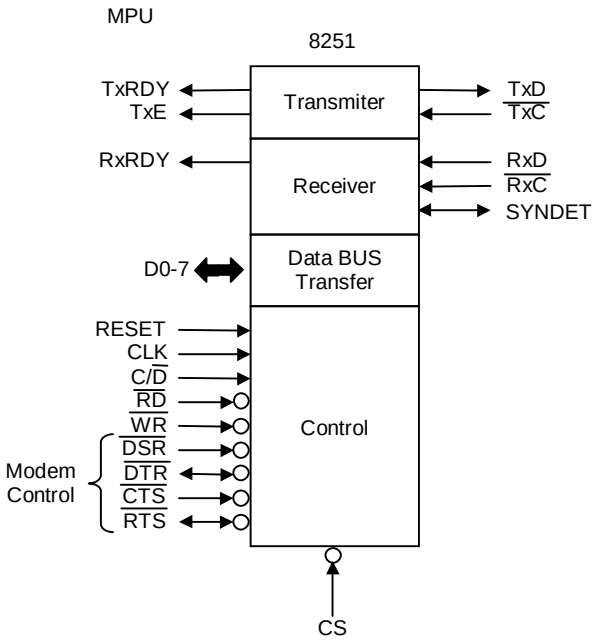


Figura 4.27. USART 8251 da INTEL.

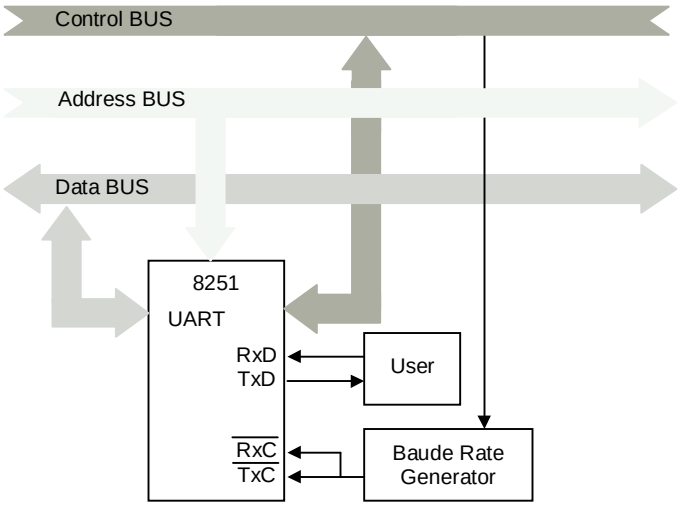


Figura 4.28. Aplicação típica de uma USART.

Linha	Função
CS	Chip Select (device)
SYNDET	Synchronous mode
RxC	Receiver clock
RxD	Receiver data (serial in)
TxC	Transmitter clock
TxD	Transmitter data (serial out)
TxDRY	Transmitter ready (to accept data)
TxE	Transmitter empty
RxRDY	Receiver ready (character ready)
DO-D7	Data bus
RESET	Resets device to idle mode
CLK	Internal clock
C/D	During read selects status or data to be input to CPU
	During write interprets data as command or data
RD	UART gates status or data on data bus
WR	UART accept command or data from data bus
Modem control	
DSR	Data set ready
DTR	Data terminal ready
CTS	Clear to send
RTS	Request to send

Figura 4.29. Sinais de controle da USART 8251.

C/D	RD	WR	CS	Operação
0	0	1	0	8251 para *Data Bus* (Read)
0	1	0	0	*Data Bus* para 8251 (Write)
1	0	1	0	*Status* para *Data Bus*
1	1	0	0	*Data Bus* para *Control*
—	—	—	1	*Data Bus* para *Tri-State*

Figura 4.30. Comandos da USART 8251.

- Operação de Saída:
  - A operação da USART é iniciada através de uma palavra de controle que seta o modo de transmissão (síncrono ou assíncrono).
  - O próximo byte seta o tamanho da palavra e bits de controle de transmissão adicionais.
  - O tamanho da palavra a ser transmitida pode ser de 5 a 8 bits.
  - A transferência é iniciada através de um procedimento de *hand-shake*. O receptor precisa sinais de temporização, providos por um relógio externo. Estes sinais são sincronizados com os bits transmitidos, de modo a o receptor poder reconhecê-los.
  - O relógio externo é fornecido pelo gerador de taxa de baud e deve ser ajustado para o modo de operação do periférico.
  - *Hand-shake*:
    - Computador ativa DTR, informando que está ligado.
    - Em resposta, periférico ativa DSR.
    - Quando o computador deseja enviar dados, ativa RTS.
    - Periférico em condição de receber dados ativa CTS.
  - Computador escreve dado no 8251.
  - Módulo de transmissão serializa o dado, anexando automaticamente um bit de inicio, dois bits de parada e um bit de paridade.
  - O receptor analisa os bits que chegam e checa a paridade, convertendo o dado serial para a forma paralela.
  - Para receber dados, a interface USART faz um *polling* da entrada de dados serial a uma taxa 16 vezes mais rápida do que a taxa de transmissão.
  - Quando um nível baixo é detectado, o *hardware* espera oito ciclos de *polling* para verificar se a linha está em nível baixo após meio período de bit. Caso afirmativo, um bit é lido a cada 16 ciclos de *polling*.
  - Ao final do byte, checa-se se bit de parada está em nível alto. Se o byte é recebido sem problemas, um bit de *flag* é setado ou o software de aplicação é interrompido.
  - A palavra recebida é armazenada em um registrador do receptor para posterior processamento pelo periférico.

- Entrada/Saída Mapeada em Memória:

  - Registradores do dispositivo de E/S recebem um endereço dentro do espaço de endereçamento do processador.
  - Para acessar o dispositivo de E/S, o processador utiliza as mesmas instruções usadas para referenciar a memória principal.
  - Não é necessário disponibilizar instruções específicas para E/S.
  - Instruções que endereçam o dispositivo de E/S colocam o endereço do mesmo na via de endereços.
  - O endereço colocado na via de endereços precisa ser decodificado para ativar o *Chip Select* do dispositivo de E/S correspondente.
  - Exemplo:
    - Processador embarcado, dedicado para controlar eletrodoméstico. Utiliza endereços de 16 bits.
    - Memória EPROM de 2 K × 8 bytes. Armazena programa de controle.
    - Memória RAM de 2 K × 8 bytes. Armazena dados e variáveis usadas pelo programa.
    - PIO com quatro registradores de 8 bits.
    - Mapeamento em Memória:
      - Endereços da EPROM: 0000 0xxx xxxx xxxx.
      - Endereços da RAM: 1000 0xxx xxxx xxxx.
      - Endereços da PIO: 1111 1111 1111 11xx.

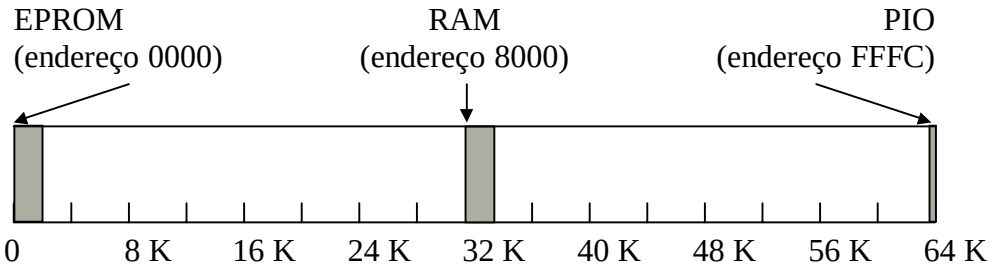


Figura 4.31. EPROM de 2 K, RAM de 2 K e PIO de 4 bytes mapeadas em espaço de endereçamento de 64 K.

- Decodificação parcial de endereços:
  - Endereços da EPROM: 0xxx xxxx xxxx xxxx.
  - Endereços da RAM: 10xx xxxx xxxx xxxx.
  - Endereços da PIO: 11xx xxxx xxxx xxxx.
- *Hardware* simples, mas não permite expansão futura.

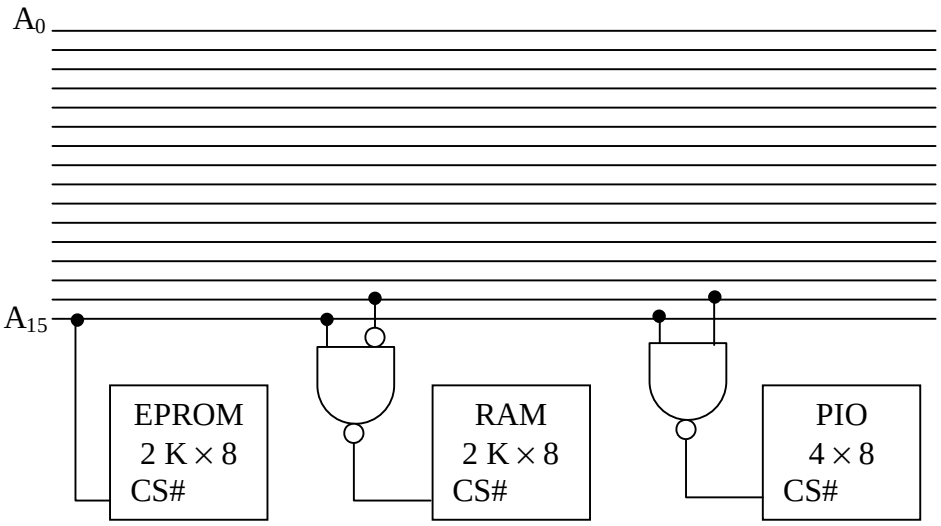


Figura 4.32. Decodificação parcial de endereços.

- Decodificação completa de endereços:
  - Endereços da EPROM: 0000 0xxx xxxx xxxx.
  - Endereços da RAM: 1000 0xxx xxxx xxxx.
  - Endereços da PIO: 1111 1111 1111 11xx.
- *Hardware* mais complexo, mas permite expansão futura.

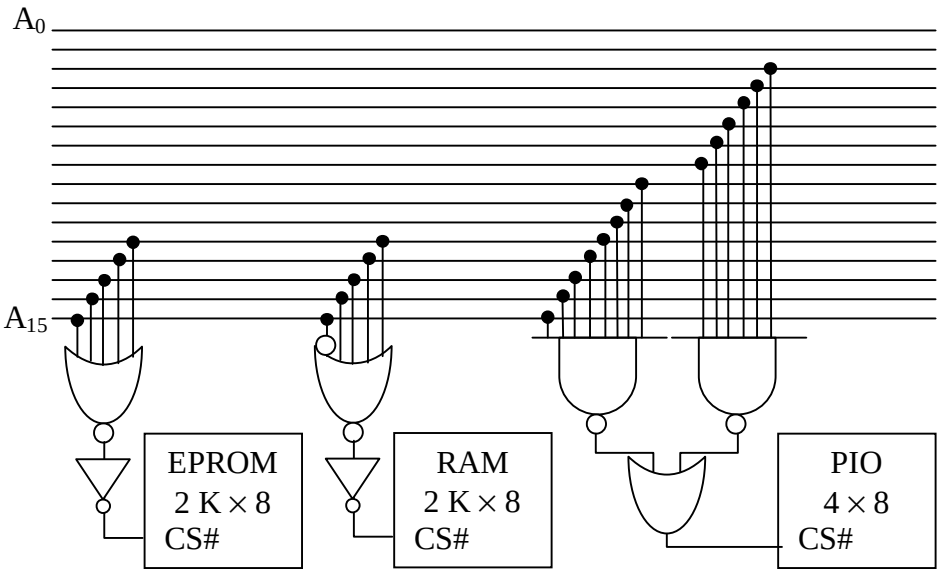


Figura 4.33. Decodificação completa de endereços.

4.6 Unidade de Processamento Central (CPU)

- *Chip* Processador:

  - CPU em um *chip*.
  - Pinos de um processador:
    - Pinos de dados (entrada e saída): conectados à via de dados. Um processador com n pinos de dados pode ler ou escrever uma palavra de n bits numa única operação. Valores típicos: n = 8, 16, 32, 36 e 64.
    - Pinos de endereço (saída): conectados à via de endereços. Um processador com m pinos de endereço pode endereçar até 2^m posições de memória. Valores típicos: n = 16, 20, 32 e 64.
    - Pinos de controle (entrada ou saída): conectados à via de controle.
      - Controle de barramento (entrada ou saída): comandam operações de leitura e escrita.
      - Arbitragem de barramento (entrada ou saída): necessários para regular o acesso ao barramento e evitar conflitos quando dois ou mais dispositivos disputam a posse do mesmo.
      - Interrupções (entrada): usados por dispositivos de E/S para sinalizar pedidos de serviço de interrupção.
    - Outros: Vcc, terra, *clock*, *status*, Reset, sinalização para co-processador, etc.

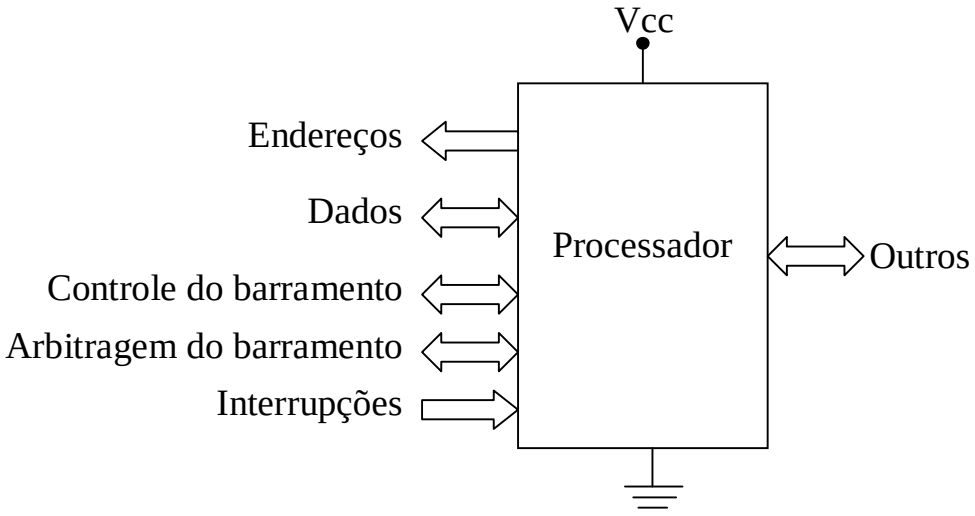


Figura 4.34. Pinagem lógica de um processador.

- Caminho de Dados:

  - Caminho lógico de processamento de dados, no qual a informação flui pelas vias internas da CPU em três estágios:
    - A informação a ser processada é lida na Memória de rascunho e disponibilizada na entrada da ULA.
    - A informação é processada na ULA.
    - A informação processada é armazenada na memória de rascunho.
  - Os três estágios acima constituem um ciclo de máquina (que geralmente é executado num ciclo de relógio)
  - O seqüenciamento do ciclo de máquina é comandado pela Unidade de Controle, responsável pela geração, na seqüência apropriada, dos sinais de controle dos circuitos lógicos que compõem o caminho de dados.

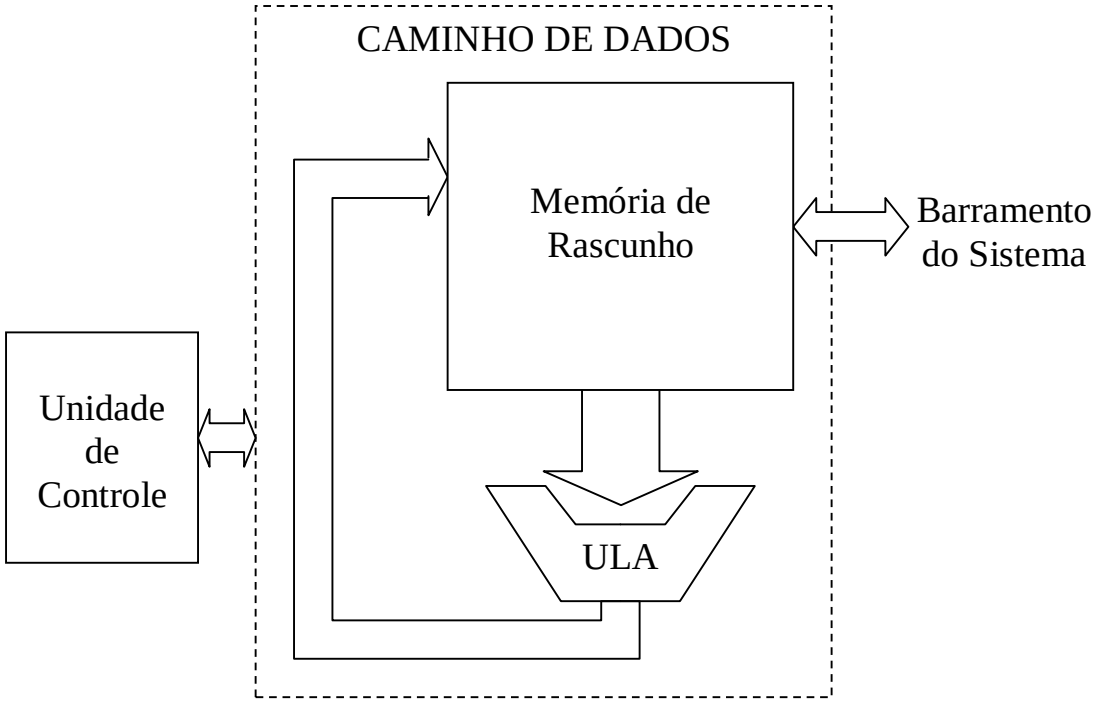


Figura 4.35. Caminho de dados de uma CPU.

- Memória de Rascunho:

  - Conjunto de m registradores (tipicamente, m = algumas dezenas), os quais constituem posições de memória internas da CPU.
  - Armazenam informações referentes à interpretação de instruções do Nível ISA.
  - Muitos são dedicados a uma função específica (PC, SP, IR, ACC, etc.), embora possam existir registradores de uso geral.
  - São acessados a cada ciclo de máquina.
  - Devem ser implementados com a tecnologia mais rápida possível, pois têm influência direta no desempenho da máquina.

- Vias Internas:

  - Barramentos internos da CPU, constituídos por n linhas de dados, (sendo n o tamanho das palavras processadas pela ULA, bem como o tamanho dos registradores da memória de rascunho).
  - Poucas linhas de controle adicionais, (geralmente uma ou duas), são usadas para controlar o tráfego nas vias internas.
  - Geralmente são dedicadas, conectando dois dispositivos (registradores, *latches*, ULA ou outros módulos de processamento).
  - Geralmente unidirecionais ou, às vezes, bidirecionais *half-duplex*.
  - Saída do dispositivo para a via interna pode ser *tri-state*.
  - Exemplo: transferência entre dois registradores R1 (fonte) e R2 (destino) conectados por via dedicada.
    - Sinal OE1 (*Output Enable*) habilita saída do registrador R1.
    - Após um certo tempo de espera, o dado armazenado em R1 fica estável, disponível na via.
    - Sinal CK2 (*ClocK*) habilita escrita do dado no registrador R2.
    - Observação: a aplicação dos sinais OE1 e CK2 é realizada na seqüência correta pela unidade de controle.

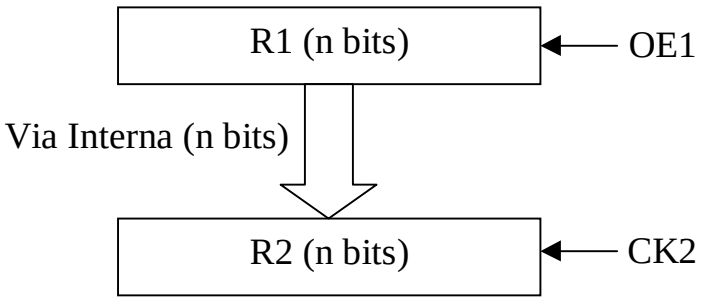


Figura 4.36. Transferência entre registradores através de via interna.

- Conexão da CPU com o Barramento Externo:

  - O acesso à memória principal geralmente é demorado (alguns ciclos de máquina), de modo que é necessário utilizar registradores para interfacear a CPU com o barramento externo.
  - Conexão com a Via de Endereços:
    - Intermediada por Registrador de Endereços da Memória (MAR - *Memory Address Register*).
    - Exemplo de Endereçamento:
      - MAR carregado a partir de via interna de saída de endereços ativando o sinal de controle CK_A.
      - Endereço armazenado em MAR é disponibilizado na via de endereços ativando o sinal de habilitação de saída (OE)do MAR, normalmente gerado como um OU lógico dos sinais de leitura e escrita (RD OR WR).
  - Conexão com a Via de Dados:
    - Intermediada por Registrador *Buffer* da Memória (MBR - *Memory Buffer Register*).
    - Exemplo de saída de dados para o barramento:
      - MBR carregado a partir de via interna de saída de dados ativando o sinal de controle CK_B.
      - Dado armazenado em MBR é disponibilizado na via de dados ativando o sinal WR (*WRite*).
    - Exemplo de entrada de dados a partir do barramento:
      - MBR carregado a partir da via de dados do barramento através de sinal RD (*ReaD*).
      - Dado armazenado em MBR é disponibilizado na via interna de entrada de dados, a qual está sempre habilitada.

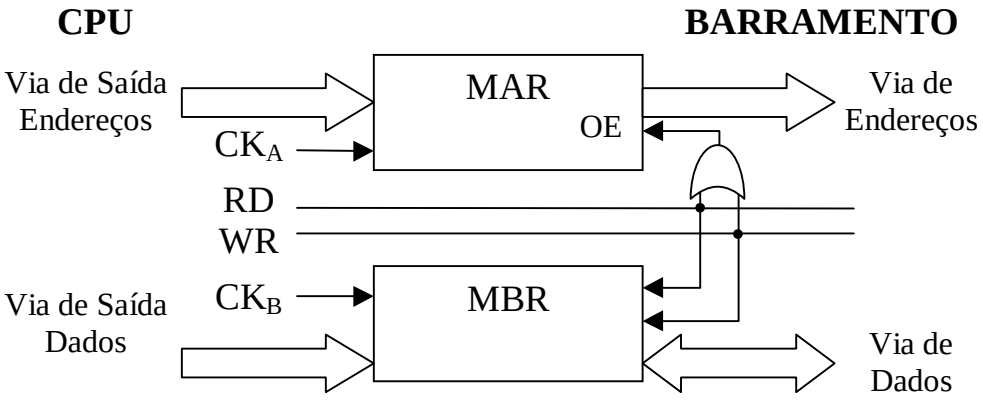


Figura 4.37. Interface entre a CPU e o Barramento.

- Unidade Lógica Aritmética:

  - Circuito combinacional que implementa funções lógicas e aritméticas sobre as entradas.
  - Entradas: duas palavras, A e B, de n bits cada uma.
  - Entrada de seleção de função: palavra F, código de m bits que seleciona uma dentre $k = 2^m$ funções disponíveis na ULA para ser aplicada sobre as entradas A e B.
  - Saída: uma palavra f(A, B), de n bits, função lógica ou aritmética das palavras de entrada A e B, de acordo com o código de seleção de função F.
  - Saída de *Status*: CC, Códigos de Condição, palavra cujos bits refletem o estado do resultado da operação realizada pela ULA. Geralmente, os bits de condição são armazenados na PSW (*Program Status Word*). Exemplo de bits de condição: Z (Zero), N (Negativo), C (*Carry*), V (*oVerflow*), A (*Auxiliar carry*), P (Paridade par), etc.

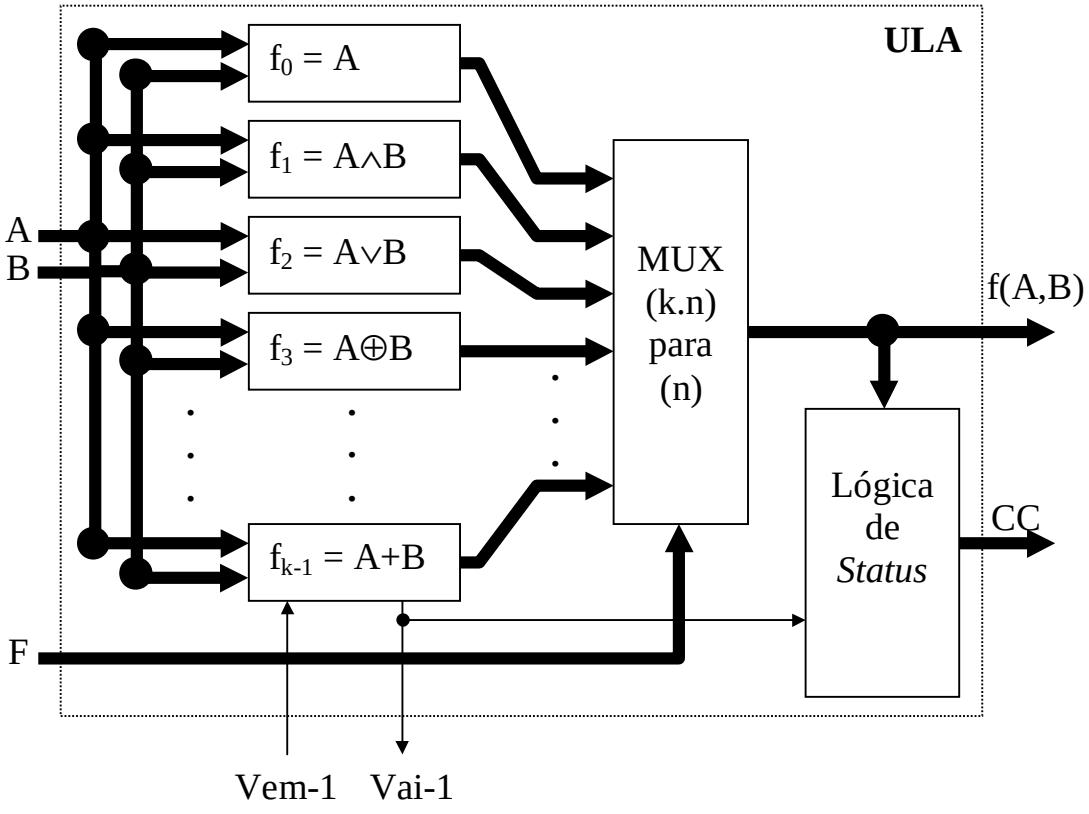
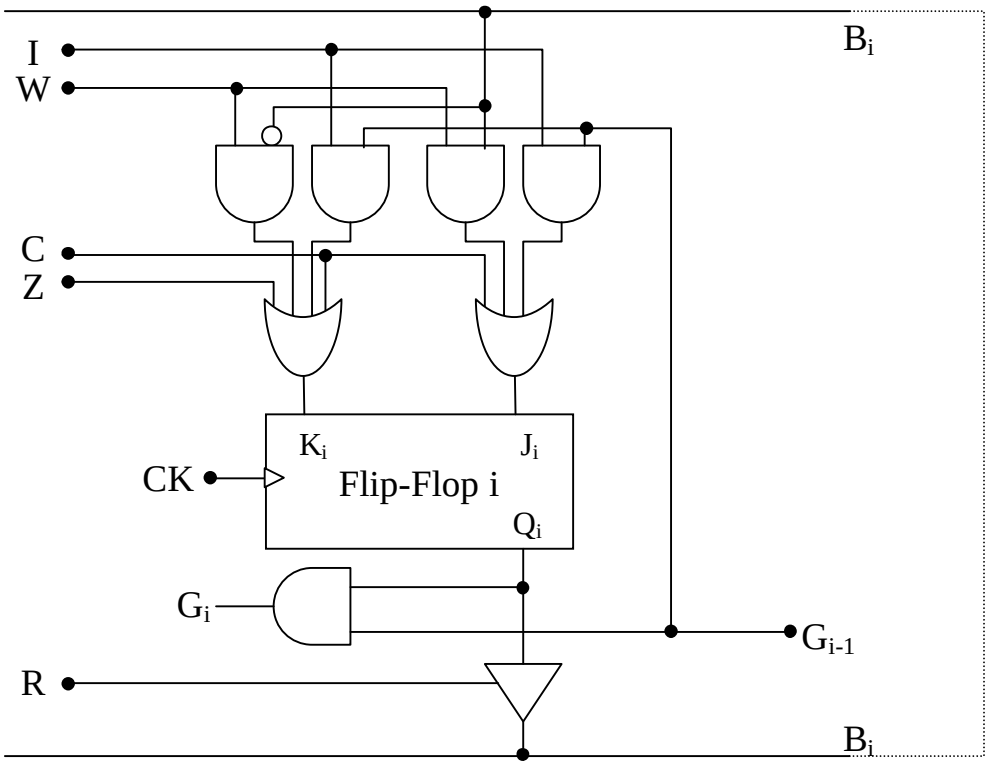


Figura 4.38. Unidade Lógica Aritmética.

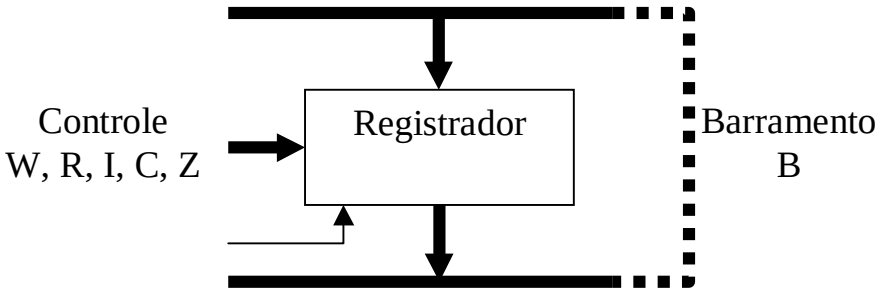
- Unidade de Controle:

  - Circuito seqüencial que gera os sinais de controle do caminho de dados na seqüência apropriada para implementar um ciclo de máquina.
  - Entradas: Instrução + *status* (além do relógio).
  - Saídas: sinais de controle do caminho de dados.
  - Implementada em *hardware* ou em *firmware* (máquinas microprogramadas).

  - Exemplo de registrador sensível a múltiplos comandos:



(a)



(b)

Figura 4.39. Registrador controlado: a) Estágio. b) Esquema funcional.

- Operações no registrador controlado:

Operação	$J_i = B_i.W+G_{i-1}.I+C$	$K_i = B_i'.W+G_{i-1}.I+C+Z$	Q_i	B_i
W (Write)	$B_i.W$	$B_i'.W$	B_i	B_i
R (Read)	0	0	Q_{i0}	Q_i
I (Incr.)	$G_{i-1}.I$, (com $J_0 = 1.I$)	$G_{i-1}.I$, (com $K_0 = 1.I$)	$(I.G_{i-1})\oplus Q_{i0}$	B_i
C (Compl.)	C	C	Q_{i0}'	B_i
Z (Zero)	0	Z	0	B_i

  - Observações:
    - Apenas um dos sinais de controle (W, R, I, C, ou Z) deve estar ativo de cada vez, (ou todos desativados).
    - Q_{i0} = Estado anterior do flip-flop i.
    - X' = not X

- <u>Exemplo de caminho de dados simples</u>:

  - "Memória principal": Registradores RX e RY, de n bits, com controle de escrita (W_x, W_y) e leitura (R_x, R_y).
  - "Memória de rascunho": Registrador Controlado RC de n bits com escrita (W), leitura (R), incremento (I), complemento (C) e zero (Z). Acumulador AC de n bits com leitura (R) e escrita (W).
  - "ULA": somador de n bits.

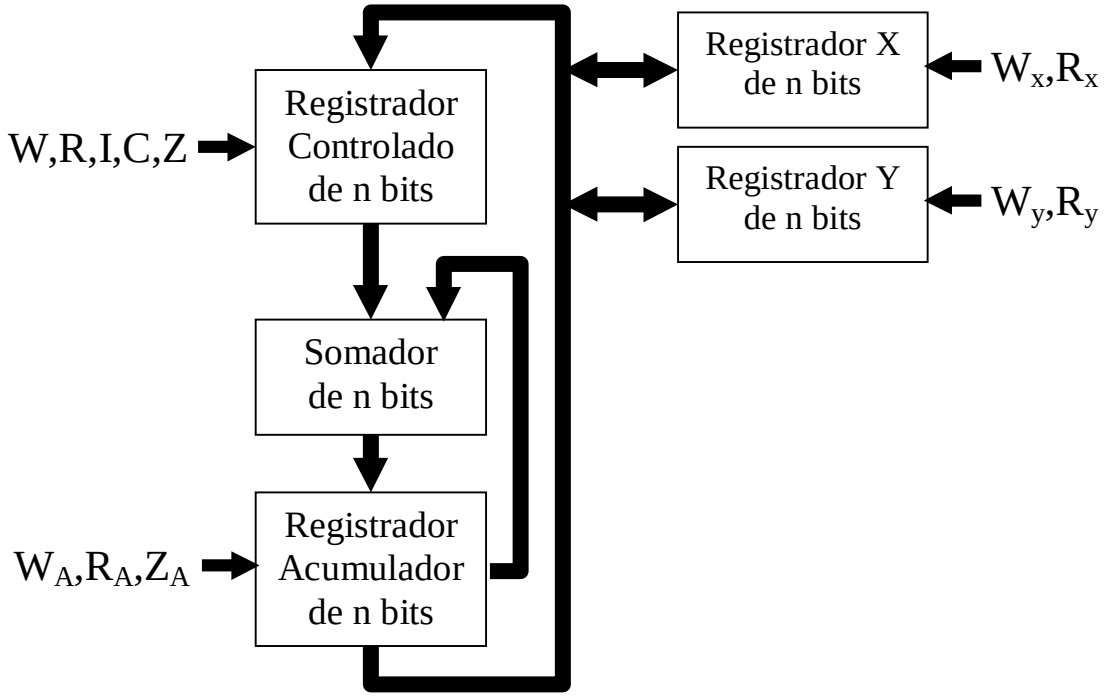


Figura 4.40. Um caminho de dados simples.

- Microoperações para controle do caminho de dados:
  - Para somar os conteúdos x e y de RX e RY e armazenar o resultado em RX, é necessário executar uma seqüência adequada de microoperações no caminho de dados.
  - Cada microoperação é executada em um ciclo de relógio.

Ciclo de *clock*	Linhas de controle a ativar	Comentário
1	Z_A, Z	AC ← 0; RC ← 0.
2	R_x, W	RC ← RX. (RC = x).
3	R, W_A	AC ← AC + RC. (AC = 0 + x = x).
4	R_y, W	RC ← RY. (RC = y).
5	R, W_A	AC ← AC + RC. (AC = x + y).
6	R_A, W_x	RX ← AC. (RX = x+y).

Figura 4.41. Microoperações para adição no caminho de dados.

- Implementação do controlador do caminho de dados:
  - O controlador do caminho de dados necessário para realizar a soma de RX e RY deve ser um circuito seqüencial capaz de gerar a seqüência de sinais de controle especificada.
  - Entradas: CK (relógio) e S (sinal de partida). O caminho não processa informação enquanto o sinal de partida não for ativado, ficando em estado de espera. Quando S é ativado, o acumulador é zerado. Após isto, S é desativado e a soma é realizada.
  - Saídas: sinais de controle Z_A, R_A, W_A, R_x, R_y, R, W, na seqüência especificada.
  - Existem sete estados: o estado de espera e os seis estados pelos quais o sistema deve passar para gerar os diferentes sinais de controle.
  - Um mínimo de três flip-flops é necessário para implementar uma máquina de estados capaz de gerar os sete estados possíveis.
  - Uma lógica adicional de decodificação é necessária para gerar os sinais de controle a partir dos estados armazenados em flip-flops.

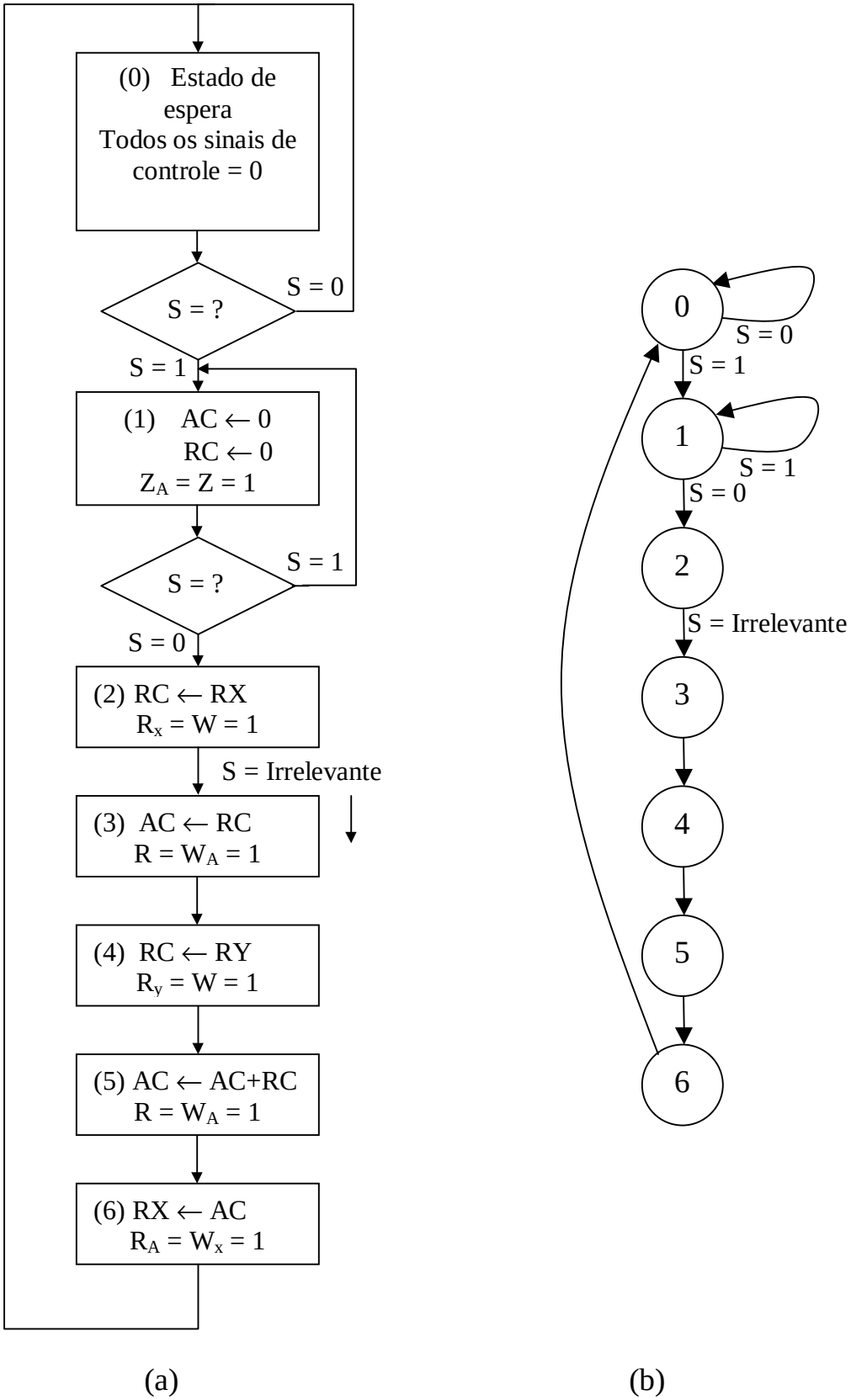


Figura 4.42. a) Fluxograma do controlador para somar RX e RY e armazenar em RX. b) Diagrama de estados correspondente.

Estado presente		Estado seguinte		Saídas								
Nº	$Q_2Q_1Q_0$	S=0	S=1	Z	Z_A	R_1	W	R	W_A	R_2	W_1	R_A
0	000	000	001	0	0	0	0	0	0	0	0	0
1	001	010	001	1	1	0	0	0	0	0	0	0
2	010	011	011	0	0	1	1	0	0	0	0	0
3	011	100	100	0	0	0	0	1	1	0	0	0
4	100	101	101	0	0	0	1	0	0	1	0	0
5	101	110	110	0	0	0	0	1	1	0	0	0
6	110	000	000	0	0	0	0	0	0	0	1	1

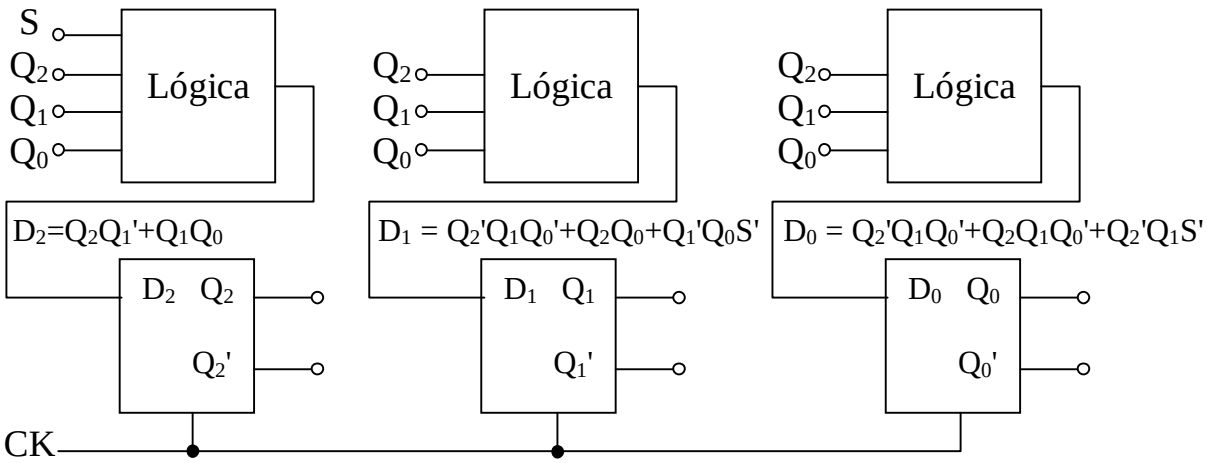
(a)

Q_0S \ Q_2Q_1	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	0	1	x	1
10	0	1	x	1

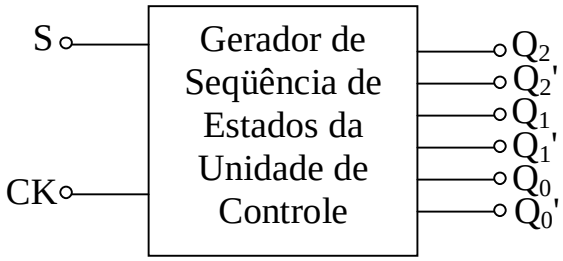
Q_0S \ Q_2Q_1	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	0	x	1
10	1	0	x	1

Q_0S \ Q_2Q_1	00	01	11	10
00	0	1	0	1
01	1	1	0	1
11	1	0	x	0
10	0	0	x	0

(b)



(c)



(d)

Figura 4.43. Projeto da Unidade de Controle. a) Tabela de transição de estados. b) Mapa K de excitação. c) Diagrama lógico. d) Bloco controlador.

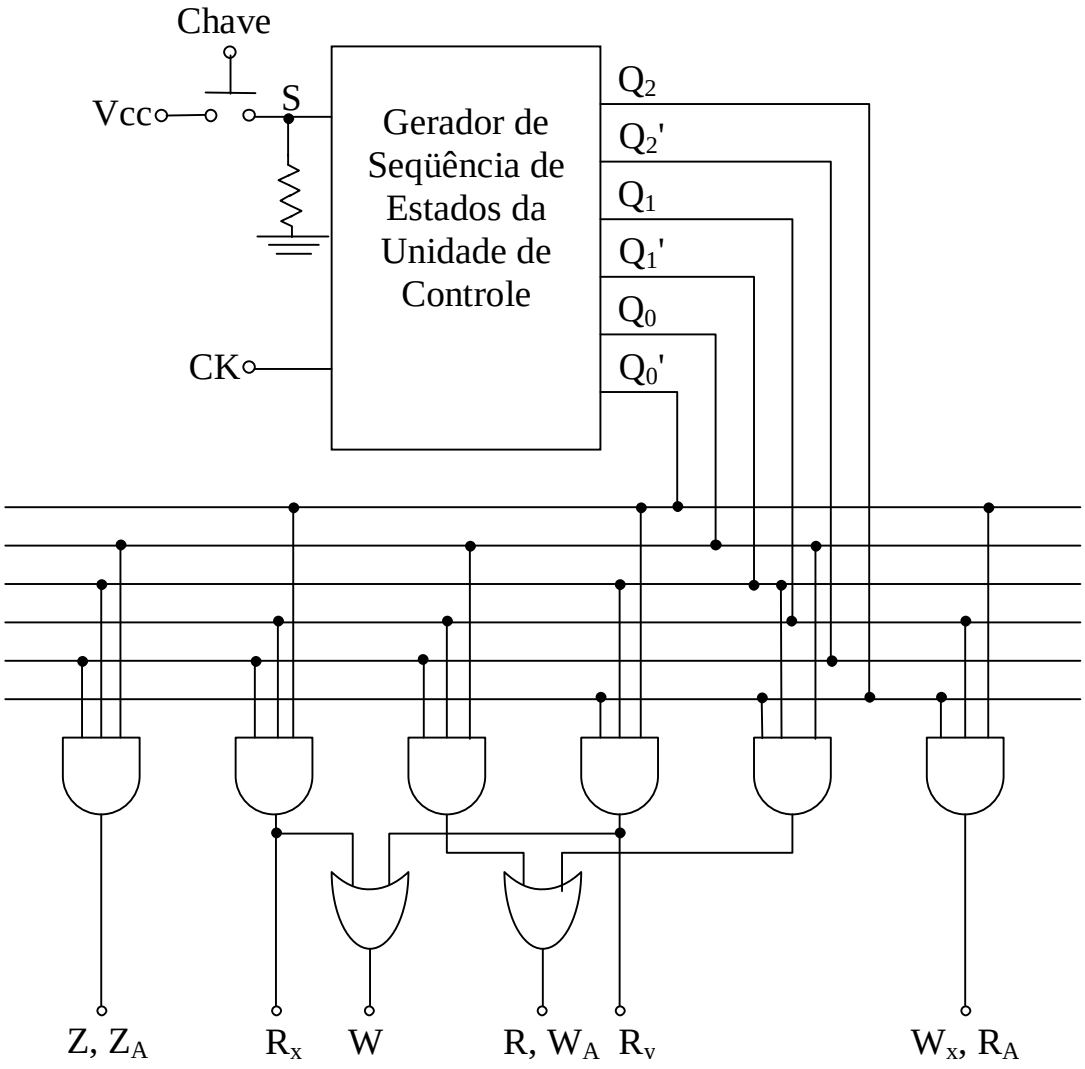


Figura 4.44. Decodificador para o gerador de seqüências do controlador.

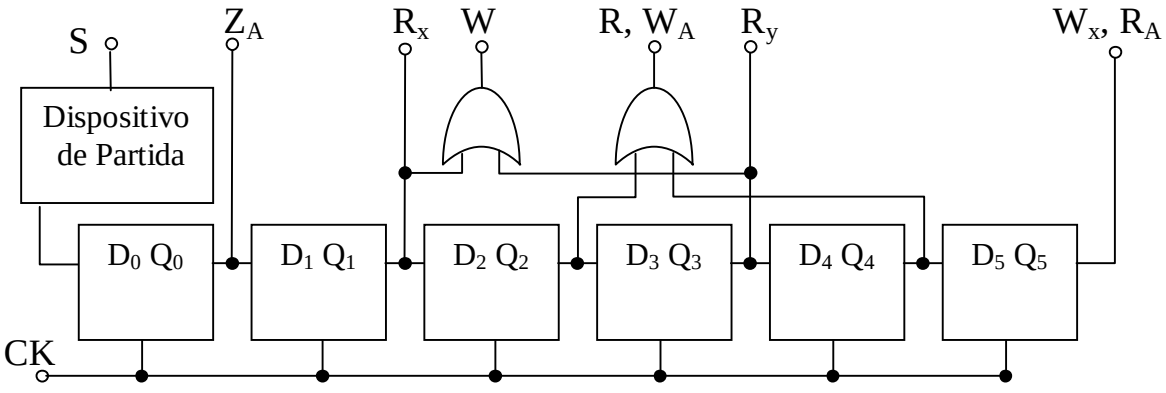


Figura 4.45. Uma outra implementação da Unidade de Controle, baseada em registrador de deslocamento. (Não mínimo, mas menos lógica).

- Unidade de Controle com resposta condicional:

  - Uma Unidade de Controle deve gerar diferentes seqüências de estados (diferentes ciclos de máquina), de modo a poder executar diferentes microinstruções.
  - Diferentes ciclos de máquina devem ser selecionados através de sinais adequados. A Unidade de controle de possuir entradas adicionais para seleção da seqüência de estados a ser gerada.
  - Exemplo: no caminho de dados simples apresentado anteriormente, um código de operação de dois bits, C_1C_0, poderia ser utilizado para selecionar diferentes ciclos de máquina:

Instrução		Operação resultante
C_1	C_0	
0	0	$AC \leftarrow R_x + R_y$
0	1	$AC \leftarrow -R_x + R_y$
1	0	$AC \leftarrow R_x - R_y$
1	1	$AC \leftarrow -R_x - R_y$

Figura 4.46. Código de operação C_1C_0 para seleção de quatro instruções.

  - Com base nesta idéia, uma Unidade de Controle para o caminho de dados simples, utilizado como exemplo, poderia ser construída a partir de quatro registradores de deslocamento diferentes, responsáveis, cada um, por um ciclo de máquina diferente.

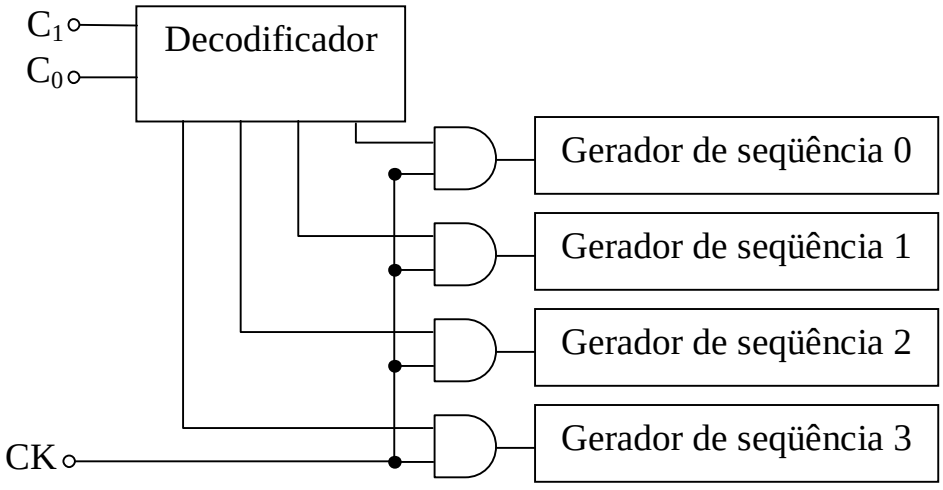


Figura 4.47. Uma Unidade de Controle que permite quatro diferentes seqüências de comprimento variável, dependendo de C_1C_0.

- Exemplo de computador simples:
  - Memória principal:
    - RAM de 64 palavras de oito bits cada.
    - Endereçada através de endereços de seis bits.
    - Sinais de controle: E (*Enable*), R/W# (Leitura/Escrita).
    - Entrada e Saída através de uma Via de Dados de oito bits.
  - Caminho de Dados simples:
    - "ULA" = somador de oito bits.
    - Registrador Acumulador (AC - 8 bits).
      - Alimentado a partir do somador (sinal W_A (escrita)).
      - Saída para a via de dados (sinal R_A (leitura)).
      - Saída para uma das entradas do somador sempre habilitada.
    - Registrador Controlado (RC - 8 bits).
      - Alimentado pela via de dados através do sinal W (escrita).
      - Saída para o somador através do sinal R (leitura).
      - Sinal I (incrementa RC). Sinal C (complementa RC).
  - Registrador de instruções (IR - 8 bits).
    - Campos:
      - Código de operação (2 bits). Define operação a ser executada pela Unidade de Controle.
      - Endereço (6 bits). Operando usado para endereçar a RAM.
    - Alimentado a partir da via de dados através do sinal TB (Transferir do Barramento).
    - Saída para a Unidade de Controle: dois bits do campo de código de operação. Sempre habilitada.
    - Saída para o Registrador de Endereços da Memória (MAR): seis bits do campo de endereço. Sempre habilitada.
  - Registrador Contador de Programa (PC - 6 bits):
    - Saída para o IR sempre habilitada.
    - Incrementado através do sinal IPC (Incrementa PC).
  - Registrador de Endereços da Memória (MAR - 6 bits):
    - Saída para a Via de Endereços sempre habilitada.
    - Alimentado pelo PC através do sinal TPC (Transferir do PC).
    - Alimentado pelo campo de endereços de IR a partir do sinal TIR (Transferir do IR).
  - Unidade de Controle:
    - Circuito seqüencial que, a partir do *clock* e do campo de código de operação do IR, gera a seqüência dos sinais de controle IPC, TPC, TIR, TB, E, R/W#, R_A, W_A, C, I, R, W.

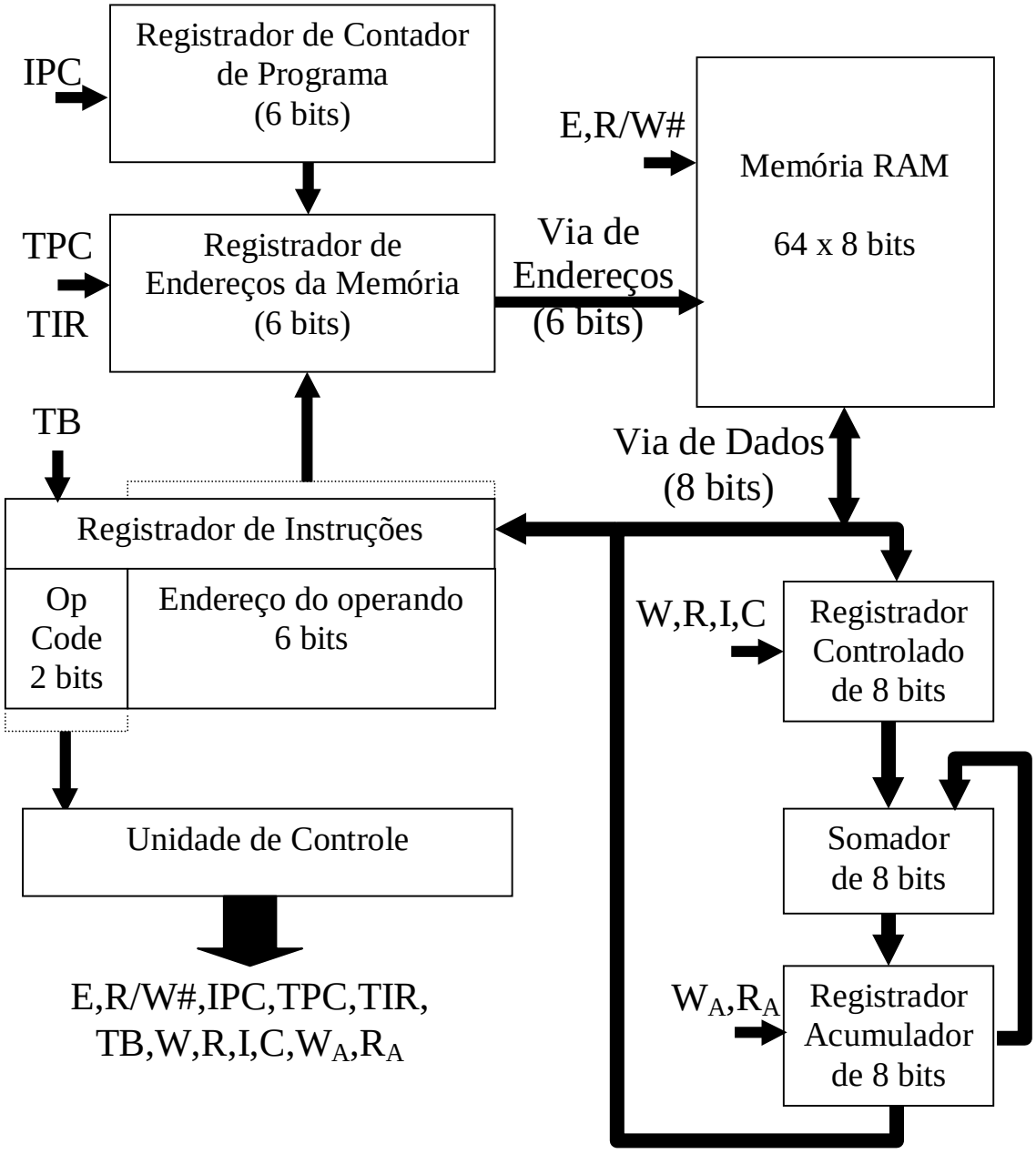


Figura 4.48. Arquitetura de um computador simples para combinação aritmética de um grande número de operandos.

- Formato de Instrução: | OPCODE (2 bits) | Endereço (6 bits) |

OPCODE	Instrução
00	Parar (STOP)
01	Somar conteúdo de Endereço a AC (ADD END)
10	Subtrair conteúdo de Endereço de AC (SUB END)
11	Transferir de AC para Endereço (STORE END)

Figura 4.49. Códigos de Operação das Instruções.

Endereço (decimal)	Conteúdo	Endereço (binário)	Conteúdo (binário)
0	SUB 59	0 0 0 0 0 0	1 0 1 1 1 0 1 1
1	ADD 60	0 0 0 0 0 1	0 1 1 1 1 1 0 0
2	SUB 61	0 0 0 0 1 0	1 0 1 1 1 1 0 1
3	ADD 62	0 0 0 0 1 1	0 1 1 1 1 1 1 0
4	ADD 63	0 0 0 1 0 0	0 1 1 1 1 1 1 1
5	STORE 39	0 0 0 1 0 1	1 1 1 0 0 1 1 1
6	STOP	0 0 0 1 1 0	0 0 x x x x x x
. . .	. . .	. . .	. . .
59	49	1 1 1 0 1 1	0 0 1 1 0 0 0 1
60	-79	1 1 1 1 0 0	1 0 1 1 0 0 1 1
61	-52	1 1 1 1 0 1	1 1 0 0 1 1 0 0
62	121	1 1 1 1 1 0	01 1 1 1 0 0 1
63	82	1 1 1 1 1 1	0 1 0 1 0 0 1 0

Figura 4.50. Conteúdo da memória, com um programa que resultará no valor - (49) + (-79) - (-52) + (121) + (82) = +127 para o endereço 39.

Ciclo de Relógio	Descrição mnemônica	Linha de Controle a habilitar
CICLO DE BUSCA (*FETCH CYCLE*)		
1. Transferir conteúdo do PC para o MAR.	MAR←PC	TPC
2. Transferir instrução endereçada para IR e incrementar o PC.	IR←[MAR] PC←PC+1	E,R/W#,TB IPC
CICLO DE EXECUÇÃO		
3. Transferir endereço no IR para o MAR.	MAR←IR	TIR
4. Transferir palavra endereçada para RC.	BUS←[MAR] RC←BUS	E,R/W#,W
5. Complementar RC	RC←RC'	C
6. Incrementar RC	RC←RC+1	I
7. Escrever saída do somador em AC	ACC←Adder	R,W_A

Figura 4.51. Ciclo de Busca e Execução para uma instrução de subtração.

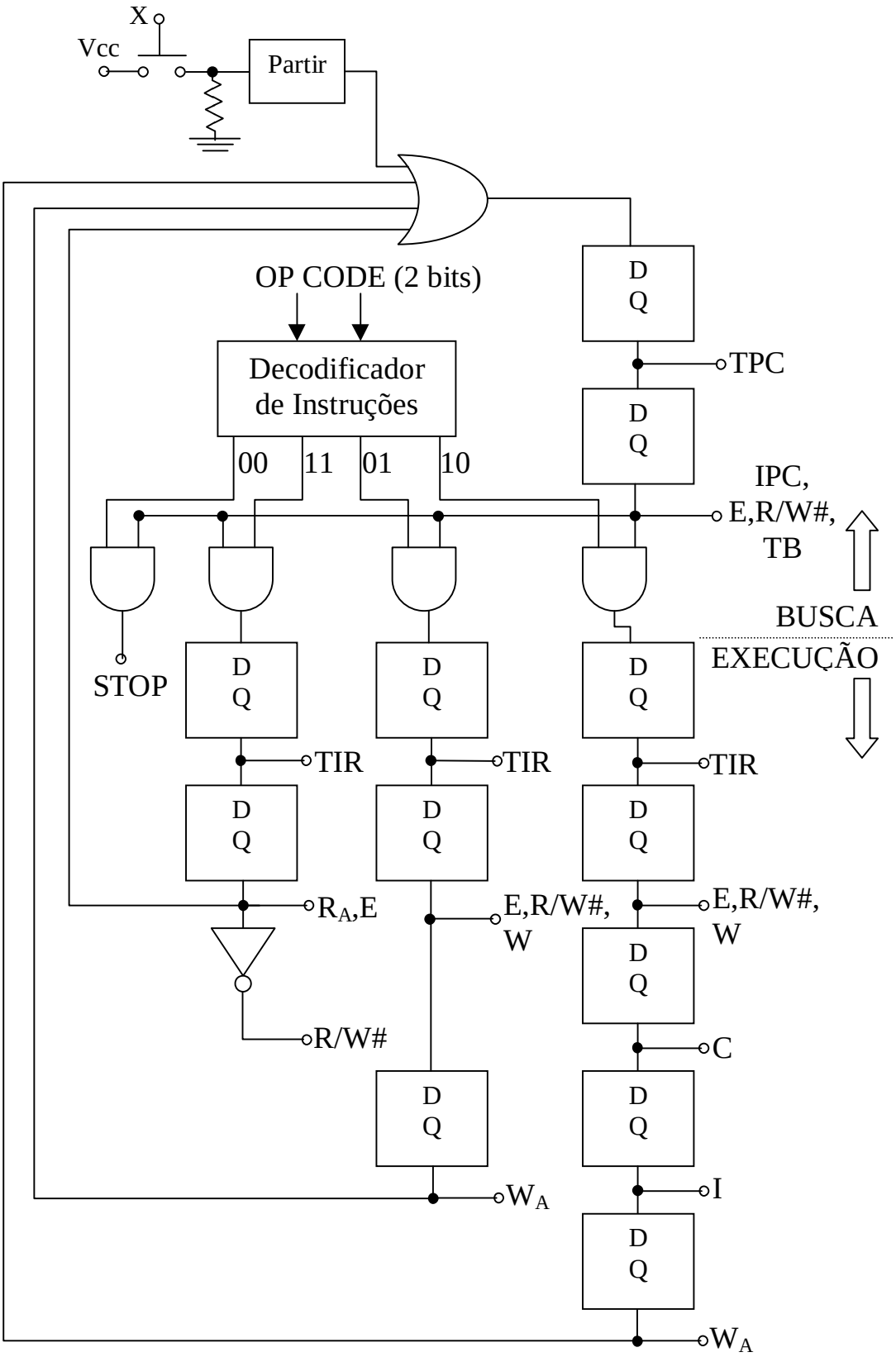


Figura 4.52. Unidade de Controle do computador simples.

4.7 Exemplo de Computador Simples

ARQUITETURA DO NÍVEL DE LINGUÁGEM DE MÁQUINA:

- Organização da memória principal: RAM de 32 posições de 8 bits.

- Tamanho da palavra: 8 bits.

- Registradores da memória de rascunho visíveis ao programador:

  - **AC**: registrador Acumulador de 8 bits. É operando implícito (fonte e destino) para algumas instruções.
  - **CP**: registrador Contador de Programa de 5 bits. Ponteiro que aponta para o endereço da memória principal onde está armazenada a próxima instrução a ser interpretada.

- Formato das Instruções:

  - Tamanho fixo de 8 bits, com Código de operação (C.O.) de 3 bits e Campo de endereço de Operando (OP.) de 5 bits.

C.O. (3 bits)	OP. (5 bits)

  - Obs.: uma instrução (STOP) não possui operando (C.O. de 8 bits).

- Conjunto de Instruções:

Mnemônico	Instrução (C.O + OP.)	Operação realizada
LOAD X	000xxxxx	AC ← (X)
STORE X	001xxxxx	(X) ← AC
ADD X	010xxxxx	AC ← AC + (X)
SUBD X	011xxxxx	AC ← AC – (X)
JUMPZ X	100xxxxx	Se AC = 0, então CP ← AC
STOP	101xxxxx	Para a execução

(*) X = operando de 5 bits xxxxx

- Modos de Endereçamento:

- Direto: OP. X = endereço na memória principal onde está o operando.
- Imediato: OP. X = constante de 5 bits passada na instrução de desvio condicional JUMPZ e que identifica o endereço de destino do desvio.

ARQUITETURA DO NÍVEL DE LÓGICA DIGITAL

- Memória principal de 32 posições de 8 bits cada. Conectada ao resto do sistema através do barramento.
- Barramento:
  - Via de Dados de 8 bits (via D).
  - Via de Endereços de 5 bits (via E).
  - Via de Controle 2 bits:
    - Linha H: Habilita memória.
    - Linha RW#: 1 = operação de leitura, 0 = operação de escrita.

- Caminho de Dados:

- Memória de rascunho.
- ULA alimentada a partir da memória de rascunho por duas vias de 8 bits (A e B) através de dois *Latches* de 8 bits (*Latch* A e *Latch* B).
- Via S (saída de dados) de 8 bits. Alimenta a memória de rascunho a partir da saída da ULA.
- O caminho de dados é ligado ao mundo externo através do barramento do sistema, interfaceado por dois registradores: Registrador de Endereços da Memória (RE) e Registrador de Dados da Memória (RD), conectados respectivamente à via de endereços (de 5 bits) e à via de dados (de 8 bits).

- Vias Internas:

- Via de alimentação da entrada A da ULA (Via A): via dedicada de 8 bits que conecta o registrador acumulador (AC) da memória de rascunho à entrada A da ULA, através do *Latch* A.
- Via de alimentação da entrada B da ULA (Via B): via de 8 bits que conecta a memória de rascunho à entrada B da ULA, através do *Latch* B.
- Via de Saída de Dados (Via S): via de 8 bits que alimenta a memória de rascunho a partir da saída da ULA.

- *Latches*: alimentam a ULA.

- *Latch* A: alimenta a entrada A da ULA a partir da via A (proveniente do Acumulador). Carregado através do sinal AA.
- *Latch* B: alimenta a entrada B da ULA a partir da via B (proveniente da memória de rascunho). Carregado através do sinal BB.

- Unidade Lógica Aritmética (ULA): combina duas entradas A e B, de 8 bits para fornecer uma saída S de 8 bits.
- Entrada A, 8 bits, proveniente do *Latch* A.
- Entrada B, 8 bits, proveniente do *Latch* B.
- Saída S, 8 bits, alimenta a via S. Combinação lógica ou aritmética das entradas A e B.
- Entrada de Seleção F, dois bits (F_1 e F_0), seleciona a função lógica ou aritmética a ser aplicada às entradas A e B para produzir a saída S.
- Saída de Status Z, 1 bit, ativada (Z=1) se a saída da ULA for zero (S=00000000), desativada (Z=0), caso contrário. Alimenta o registrador de status (ST) da memória de rascunho.

F_1F_0	S = F(A,B)
0 0	B
0 1	1+B
1 0	A+B
1 1	A–B

S = 00000000	Z = 1
S ≠ 00000000	Z = 0

- Memória de Rascunho – constituída por seis registradores dedicados:

- Registrador de Endereços da Memória (RE), 5 bits: endereça a memória principal através da via de Endereços. Alimenta a via de endereços (sinal REE) e os cinco bits menos significativos da via B (sinal REB). Alimentado a partir da via S (sinal SER).
- Registrador de Dados da Memória (RD), 8 bits: conecta o caminho de dados à via de dados do barramento. Alimenta a via de Dados (sinal RDD) e a via B (RDB). Alimentado a partir da Via de Dados (sinal DRD) e da via S (SRD).
- Registrador Contador de Programa (CP), 5 bits: armazena o endereço (ponteiro) da próxima instrução a ser interpretada. Alimenta os cinco bits menos significativos da via B (sinal CPB). Alimentado a partir da via S (sinal SCP).
- Registrador Acumulador (AC), 8 bits: armazena temporariamente operandos e resultados do processamento da ULA. Alimenta a via A (saída sempre habilitada) e a via B (sinal ACB). Alimentado a partir da via S (sinal SAC).
- Registrador de Instrução (RI), 8 bits: armazena a instrução corrente, buscada na memória. Possui dois campos: C.O. (3 bits mais significativos - armazenam o código de operação) e OP. (5 bits menos significativos - armazenam endereço de operando). Alimenta a via B (sinal RIB) e a Unidade de Controle (campo C.O. – saída sempre habilitada). Alimentado a partir da via S (sinal SRI).
- Registrador de Status (ST), 1 bit: armazena o estado corrente da saída da ULA (1 se S = 00000000, 0 caso contrário). Alimentado a partir da linha de saída de status Z da ULA (através do sinal ZST). Alimenta a Unidade de Controle (saída sempre habilitada).

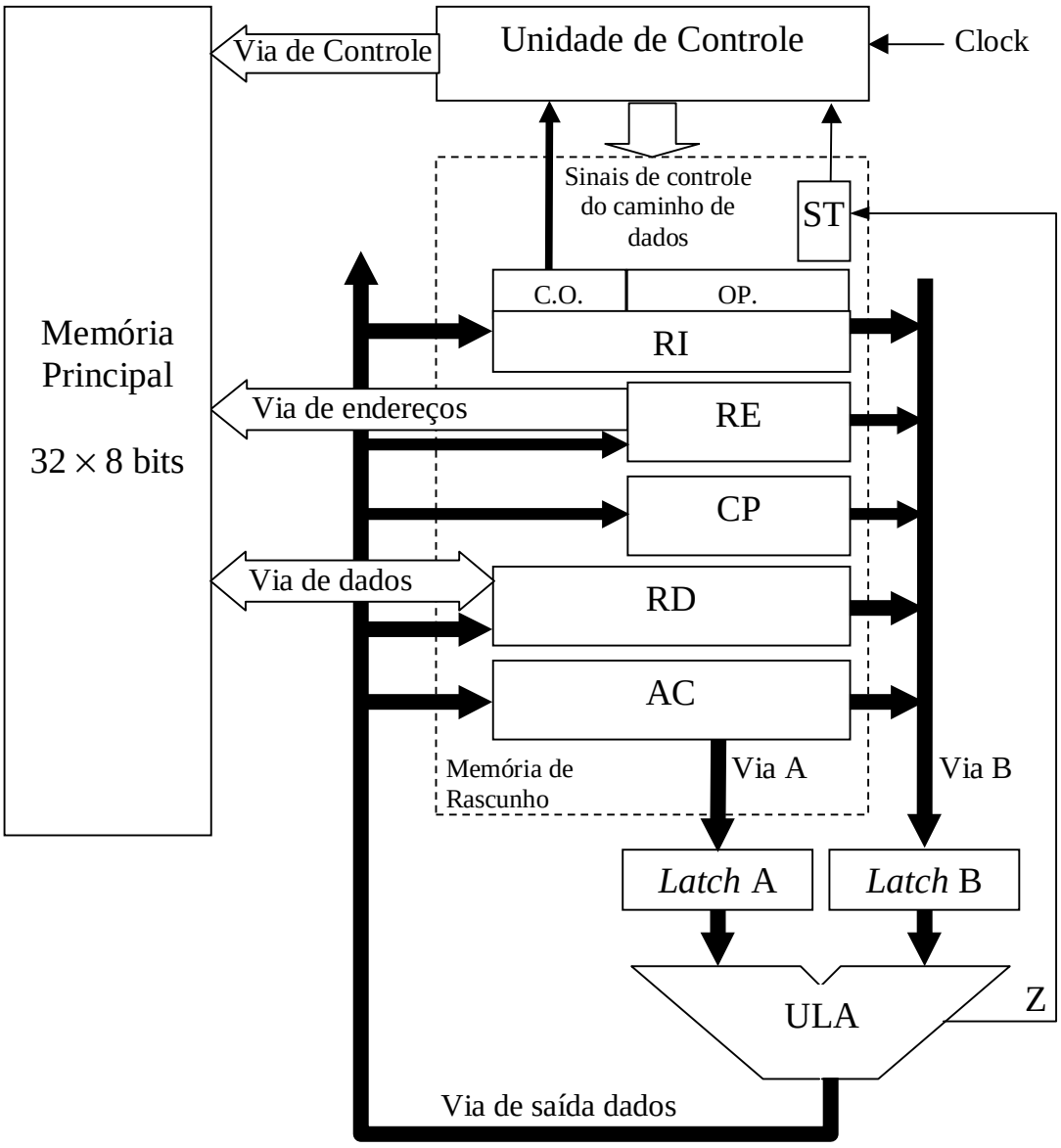


Figura 4.53. Caminho de dados do computador.

- Sinais de Controle:

Sinal	Operação resultante
H	Habilita memória RAM
RW#	0 = escrita, 1 = leitura na memória
RIB	RI → via B
SRI	via S → RI
REB	RE → via B
REE	RE → via de Endereços
SRE	Via S → RE
CPB	CP → via B
SCP	via S → CP
RDB	RD → via B
RDD	RD → via de Dados
SRD	via S → RD
DRD	via de Dados → RD
ACB	AC → via B
SAC	via S → AC
AA	AC → *Latch* A
BB	Via B → *Latch* B
ZST	Linha de Status Z → ST
F_1F_0	Sinais de Seleção da ULA: (00: B → via S) (01: 1+B → via S) (10: A+B → via S) (11: A–B → via S)

UNIDADE DE CONTROLE

Busca de Instruções:

SEQÜÊNCIA DE SINAIS PARA BUSCA DE INSTRUÇÃO		
Ciclo	Sinais	Ação resultante
1	CPB	CP → via B
2	BB	Via B → *Latch* B
3	$F_1F_0 = 00$	*Latch* B → via S
4	SRE	Via S → RE
5	REE	RE → via de Endereços
6	H RW# = 1	Habilita memória principal Leitura: (CP) → via de Dados
7	DRD	via de Dados → RD
8	RDB	RD → via B
9	BB	Via B → *Latch* B
10	$F_1F_0 = 00$	*Latch* B → via S
11	SRI	Via S → RI
12	CPB	CP → via B
13	BB	Via B → *Latch* B
14	$F_1F_0 = 01$	(*Latch* B + 1) → via S
15	SCP	Via S → CP

Decodificação de Instruções:

Hardware decodificador recebe código de operação do campo C.O. do Registrador de Instrução e ativa apenas uma dentre as seis linhas de habilitação dos seis correspondentes circuitos geradores de seqüência de sinais para execução de uma dentre as seis instruções definidas no conjunto de instruções do nível de linguagem de máquina.

Execução de Instruções:

SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **LOAD X**		
Ciclo	Sinais	Ação resultante
16	RIB	RI → via B
17	BB	Via B → *Latch* B
18	$F_1F_0 = 00$	*Latch* B → via S
19	SRE	Via S → RE
20	REE	RE → via de Endereços
21	H RW# = 1	Habilita memória principal Leitura: (OP.) → via de Dados
22	DRD	via de Dados → RD
23	RDB	RD → via B
24	BB	Via B → *Latch* B
25	$F_1F_0 = 00$	*Latch* B → via S
26	SAC	Via S → AC

SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **STORE X**		
Ciclo	Sinais	Ação resultante
16	RIB	RI → via B
17	BB	Via B → *Latch* B
18	$F_1F_0 = 00$	*Latch* B → via S
19	SRE	Via S → RE
20	ACB	AC → via B
21	BB	Via B → *Latch* B
22	$F_1F_0 = 00$	*Latch* B → via S
23	SRD	Via S → RD
24	REE RDD	RE → via de Endereços RD → via de Dados
25	H RW# = 0	Habilita memória principal Escrita: via de Dados → (OP.)

SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **ADD X**		
Ciclo	Sinais	Ação resultante
16	RIB	RI → via B
17	BB	Via B → *Latch* B
18	$F_1F_0 = 00$	*Latch* B → via S
19	SRE	Via S → RE
20	REE	RE → via de Endereços
21	H RW# = 1	Habilita memória principal Leitura: (OP.) → via de Dados
22	DRD	via de Dados → RD
23	RDB	RD → via B
24	AA BB	AC → *Latch* A Via B → *Latch* B
25	$F_1F_0 = 10$	(Latch A + Latch B) → via S
26	SAC	Via S → AC

SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **SUBD X**		
Ciclo	Sinais	Ação resultante
16	RIB	RI → via B
17	BB	Via B → *Latch* B
18	$F_1F_0 = 00$	*Latch* B → via S
19	SRE	Via S → RE
20	REE	RE → via de Endereços
21	H RW# = 1	Habilita memória principal Leitura: (OP.) → via de Dados
22	DRD	via de Dados → RD
23	RDB	RD → via B
24	AA BB	AC → *Latch* A Via B → *Latch* B
25	$F_1F_0 = 11$	(Latch A – Latch B) → via S
26	SAC	Via S → AC

SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **JUMPZ X** (1ª Parte: Teste do Acumulador)		
Ciclo	Sinais	Ação resultante
16	ACB	AC → via B
17	BB	Via B → *Latch* B
18	$F_1F_0 = 00$	*Latch* B → via S
19	ZST	Linha de status Z → ST
SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **JUMPZ X** (2ª Parte – Caso 1: ST = 1 (AC = 0))		
20	RIB	RI → via B
21	BB	Via B → *Latch* B
22	$F_1F_0 = 00$	*Latch* B → via S
23	SCP	Via S → CP
SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **JUMPZ X** (2ª Parte – Caso 2: ST = 0 (AC ≠ 0))		
20		Reinicia o ciclo de busca-decodificação-execução buscando a próxima instrução em (CP+1)

SEQÜÊNCIA DE SINAIS PARA EXECUÇÃO DE **STOP**		
Ciclo	Sinais	Ação resultante
16		Interrompe o ciclo de busca-decodificação-execução

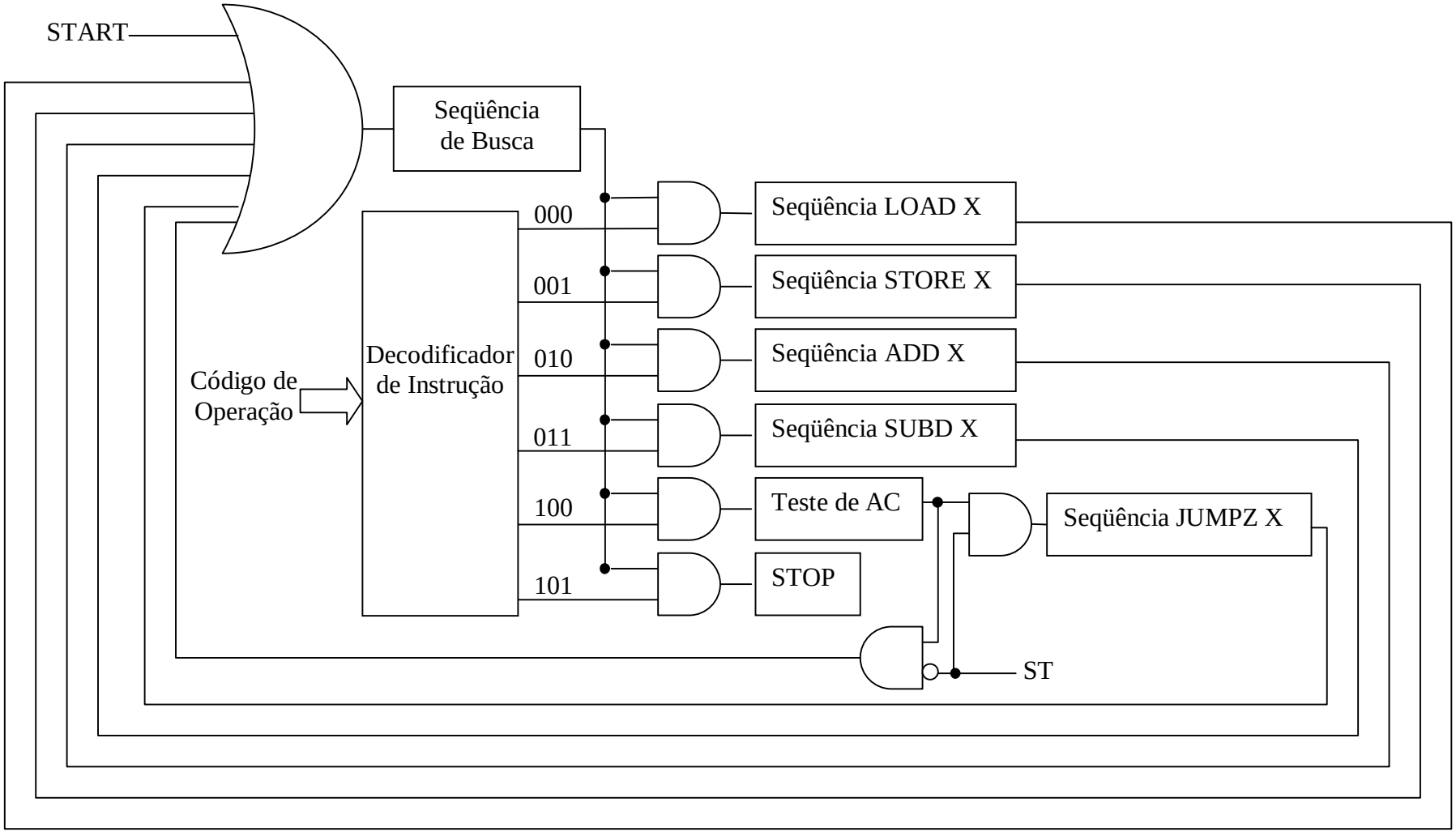


Figura 4.54. Unidade de Controle do computador.